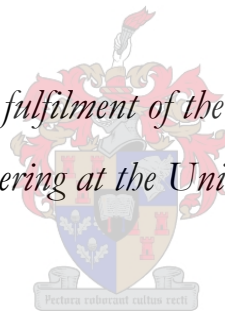


# Detecting potholes with monocular computer vision: A performance evaluation of techniques

*By*

Sonja Nienaber

*Thesis presented in partial fulfilment of the requirements for the degree Master  
of Science in Engineering at the University of Stellenbosch*



Supervisor: Dr. MJ Booysen

Department of Electrical & Electronic Engineering, Stellenbosch University

Co-supervisor: Dr. RS Kroon

Computer Science Division, Stellenbosch University

March 2016

# DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: . . . . . March 2016. . . . .

# ABSTRACT

Potholes in road surfaces create problems for motorists and driverless vehicles. This is because the damage to the vehicle that can be caused by hitting a pothole with a vehicle can be costly and even dangerous. Previous works of other authors with respect to pothole detection did not investigate the limitations of the detection capabilities of their works such as the distance at which the potholes could be detected and often used footage where the camera was directly facing the road, thereby only having a viewing range of roughly 2-4 m.

In order to complete this project, it was necessary to obtain suitable footage of potholes. The method for collecting the pothole footage can be seen as novel. The method included attaching a GoPro camera inside of a vehicle windscreen and photographing the road as the vehicle was driven around. This footage is, therefore, akin to a driver's viewpoint of the road. This viewpoint is advantageous as it ensures that the maximum amount of the road can be photographed by the camera. By mounting the camera in this manner, it could potentially be possible to detect potholes before the vehicle reaches them as opposed to other works done where the camera was mounted to the rear of the vehicle. In the instance of a driverless vehicle, this would allow the vehicle to avoid hitting the pothole and would prevent damage to the vehicle.

Due to the difficulty of detecting potholes, the footage was split into two different datasets namely, a simple scenario and a complex scenario. The simple scenario considered footage where the road lighting conditions were always open and clear. In this scenario, the road was extracted and only the extracted region was used in pothole detection algorithms. The complex scenario considered footage where the road lighting conditions were either open or contained mixed lighting conditions. Therefore, in this scenario, the input images were cropped to the suspected road region within the image. This region is rectangular and contains additional information along the sides of the image such as foliage etc.

Image processing algorithms, as well as machine learning algorithms were deployed in this thesis to investigate the feasibility of pothole detection. The machine learning algorithms used, consisted of an LBP (Local Binary Pattern) cascade classifier and an SVM (Support Vector Machine) with HOG (Histogram of Oriented Gradients) features.

The pothole locations were also analysed in terms of the relative distance that a pothole occurred from the camera. This process is known as depth estimation in monocular images, and

this work allowed for determining the ranges at which pothole detection was more successful than others. The discrepancy in the results at the different depth ranges might indicate that different algorithms and classifiers need to be implemented for different ranges to increase the performance of the pothole detector.

The final results of this project indicate that under certain conditions, it is possible to detect potholes with modest results.

# UITTREKSEL

Slaggate in die pad skep probleme vir motoriste en bestuurderlose voertuie. Dit is omdat die skade wat veroorsaak kan word aan die voertuig deur 'n slaggat te slaan duur en selfs gevaarlik kan wees. Vorige werke van ander skrywers met betrekking tot slaggat opsporing, het nie die beperkinge van die opsporing vermoëns van hul werke ondersoek nie soos byvoorbeeld, die afstand wat die slaggate kon opgespoor word en dikwels was beeldmateriaal gebruik waar die kamera direk na die pad kyk en sodoende net besigtiging van ongeveer 2-4 m het.

Ten einde hierdie projek te voltooi, was dit nodig om geskikte beeldmateriaal van slaggate te bekom. Die metode vir die insameling van die slaggat beeldmateriaal kan gesien word as nuut. Die metode behels die hegting van 'n GoPro kamera aan die binnekant van 'n voertuig voorruit en om dan die pad fotografeer soos wat die voertuig bestuur word. Hierdie materiaal is dus soortgelyk aan die oogpunt van 'n bestuurder. Hierdie metode is voordelig aangesien dit verseker dat die maksimum area van die pad gefotografeer kan word deur die kamera. Deur die kamera te monteer op hierdie wyse, kan dit potensieel moontlik wees om slaggate op te spoor voordat die voertuig hulle bereik, in teenstelling met ander werke gedoen waar die kamera aan die agterkant van die voertuig gemonteer was. In die geval van 'n bestuurderlose voertuig, sou dit die voertuig in staat stel om slaggate te vermy en gevolglik skade aan die voertuig te voorkom.

As gevolg van die probleme van die opsporing van slaggate, was die beeldmateriaal verdeel in twee verskillende datastelle naamlik 'n eenvoudige geval en 'n komplekse geval. Die eenvoudige geval oorweeg beeldmateriaal waar die pad altyd oop en duidelik was. In hierdie geval, is die pad onttrek en slegs die onttrekde streek is gebruik om slaggate in op te spoor. Die komplekse geval oorweeg beeldmateriaal waar die pad omstandighede of oop of gemengde lig omstandighede gehad het. Daarom, in hierdie geval, was die beelde geknip om die vermeende pad streek binne die beeld uit te haal. Hierdie streek is reghoekig en bevat aanvullende inligting langs die kante van die beeld soos blare en bome.

Beeldverwerking algoritmes, asook masjien leer algoritmes is ontplooi in hierdie verhandeling om die haalbaarheid van slaggat opsporing te ondersoek. Die masjien leer algoritmes wat hier bruik word, bestaan uit 'n LBP (Local Binary Pattern) kaskade klassifiseerder en 'n SVM (Support Vector Machine) met HOG (Histogram of Oriented Gradients) kenmerke.

Die slaggate is ook ontleed in terme van die relatiewe afstand wat 'n slaggat plaasgevind vanaf die kamera. Hierdie proses staan bekend as diepte skatting in monokulêre beelde. Hierdie werk het dit moontlik gemaak om te bepaal hoe suksesvol die slaggat opsporing was met

betrekking tot die afstand waar die slaggat voorkom per klassifiseerder. Die verskil in die resultate van die verskillende diepte reekse kan dui dat verskillende algoritmes en klassifiseerders nodig is vir verskillende reekse om die prestasie van slaggat opspoorders te verbeter.

Die finale uitslae van hierdie projek dui aan dat onder sekere omstandighede, is dit moontlik om slaggate op te spoor met 'n beskeie resultate.

# PUBLICATIONS

The following publications were made from the work presented in this thesis:

- S.Nienaber, M.J. Booysen, R.S. Kroon. “Detecting potholes using simple image processing techniques and real-world footage”, *2015 South African Transport Conference (SATC)*, July 2015, Pretoria, South Africa.
- S. Nienaber, R.S. Kroon, M.J. Booysen, “A Comparison of Low-Cost Monocular Vision Techniques for Pothole Distance Estimation”, *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, December 2015, Cape Town, South Africa.

Two annotated pothole datasets were created for this project and is freely available online at: <http://goog.gl/Uj38Sf>.

# ACKNOWLEDGEMENTS

I would like to thank the following people and entities for their assistance:

- My mother for her emotional support and encouragement during the *entire* duration of the degree.
- Jenna for late night phone calls of encouragement during the all-nighters.
- Dr. MJ Booysen. I would like to thank him for taking me as one of his students and giving me the opportunity to go overseas for an internship.
- Dr. RS Kroon, for his assistance throughout the year and especially the corrections to the thesis document.
- Dr. W Brink for his assistance with the depth estimation work.
- Prof. BM Herbst for his general advice with respect to machine learning as applied to computer vision.
- Prof. TR Niesler for letting me borrow one of his DSP lab computers for the duration of the project.
- Prof. K Jenkins for his insight into the physical aspects of the formation of potholes.
- MTN for providing me with financial support for the project.
- Monster energy drinks.



# TABLE OF CONTENTS

DECLARATION.....	ii
ABSTRACT.....	iii
UITTREKSEL.....	v
PUBLICATIONS.....	vii
ACKNOWLEDGEMENTS.....	viii
LIST OF FIGURES.....	iv
LIST OF TABLES.....	iii
ABBREVIATIONS AND ACRONYMS.....	iii
CHAPTER 1.....	4
INTRODUCTION.....	4
1.1        BACKGROUND.....	4
1.2        PURPOSE OF THE PROJECT.....	5
1.3        RESEARCH OBJECTIVES.....	6
1.4        SCOPE OF WORK.....	7
1.5        THESIS STRUCTURE.....	7
CHAPTER 2.....	9
LITERATURE REVIEW.....	9
2.1        INTRODUCTION.....	9
2.2        IMAGE PROCESSING.....	10
2.2.1    COLOUR SPACES.....	10
2.2.2    NOISE REMOVAL TECHNIQUES IN IMAGES.....	12
2.2.3    CANNY EDGE DETECTION.....	14
2.2.4    CONTOUR DETECTION.....	17
2.2.5    CONVEX HULL ALGORITHM.....	17
2.3        FEATURE TYPES.....	18
2.3.1    LOCAL BINARY PATTERNS.....	19

2.3.2	HISTOGRAM OF ORIENTED GRADIENTS.....	21
2.4	MACHINE LEARNING.....	22
2.4.1	INTRODUCTION .....	22
2.4.2	CASCADE CLASSIFIER .....	23
2.4.3	SVM CLASSIFIER.....	25
2.4.4	SLIDING WINDOW AND IMAGE SCALING .....	29
2.5	EXISTING LITERATURE WITH RESPECT TO POTHOLE DETECTION	
	31	
2.5.1	DETECTION OF POTHOLE VIA VIBRATIONS.....	31
2.5.2	VISUAL APPROACH.....	32
2.5.3	INFRARED APPLICATIONS IN POTHOLE DETECTION .....	37
2.6	CONCLUSION.....	39
CHAPTER 3	.....	40
DATA COLLECTION AND PREPARATION	.....	40
3.1	INTRODUCTION.....	40
3.2	DATA COLLECTION .....	40
3.3	DATA SET SELECTION .....	41
3.4	DATA PRE-PROCESSING.....	42
3.5	DATA SET ANNOTATION .....	43
3.5.1	POSITIVE SAMPLE PROCESSING .....	44
3.5.2	NEGATIVE SAMPLE PROCESSING.....	44
3.6	SAMPLE SUMMARY.....	45
3.7	DEPTH ESTIMATION.....	47
3.7.1	INTRODUCTION .....	47
3.7.2	EXISTING WORK .....	48
3.7.3	RELEVANT LITERATURE.....	49
3.7.4	METHODOLOGY .....	53

3.7.5	RESULTS OF DEPTH ESTIMATION ALGORITHMS.....	59
3.7.6	DEPTH ESTIMATION AS APPLIED TO POTHOLE? .....	62
3.7.7	DEPTH ESTIMATION SUMMARY .....	62
3.8	CONCLUSION.....	64
CHAPTER 4 .....		65
DEVELOPMENT AND METHODOLOGY .....		65
4.1	INTRODUCTION.....	65
4.2	ROAD EXTRACTION METHOD.....	65
4.3	IMAGE PROCESSING APPROACH .....	67
4.4	DETECTOR WINDOW SIZE AND HOG DESCRIPTOR PARAMETERS 69	
4.4.1	POTHOLE SIZE DISTRIBUTION .....	70
4.4.2	HOG DESCRIPTOR .....	72
4.5	SVM DEVELOPMENT.....	75
4.5.1	CROSS-VALIDATION .....	75
4.5.2	GRID SEARCH.....	76
4.5.3	SVM KERNEL SELECTION .....	78
4.6	CONCLUSION.....	78
CHAPTER 5 .....		80
RESULTS .....		80
5.1	INTRODUCTION.....	80
5.2	PERFORMANCE METRICS .....	80
5.3	MACHINE LEARNING PROGRAM SETTINGS .....	82
5.4	RESULTS OF IMAGE PROCESSING .....	83
5.5	RESULTS OF CASCADE CLASSIFIER.....	85
5.6	RESULTS OF SVM.....	87
5.7	COMPARISON OF ALL CLASSIFIERS .....	89

---

5.8	ANALYSIS OF RESULTS WITH RESPECT TO DEPTH .....	92
5.9	CLASSIFIER TIMING PERFORMANCE .....	97
5.10	CONCLUSION .....	98
CHAPTER 6 .....		100
CONCLUSION .....		100
6.1	SUMMARY OF WORK .....	100
6.2	CONCLUSIONS .....	100
6.2.1	POTHOLE DETECTION .....	101
6.2.2	DEPTH ESTIMATION TECHNIQUES .....	102
6.2.3	POTHOLE DETECTION RANGE LIMITATIONS .....	103
6.3	FUTURE WORK .....	104
<b>APPENDIX A – CASCADE CLASSIFIER SETTINGS</b> .....		105
<b>APPENDIX B – DETECTOR SETTINGS</b> .....		108
<b>APPENDIX C – PRACTICAL DEPTH ESTIMATION CONSIDERATIONS</b> .....		110
C.1 FISHEYE REMOVAL .....		110
C.2 PRACTICAL DEPTH ESTIMATION .....		111
BIBLIOGRAPHY .....		113

# LIST OF FIGURES

Figure 1.1.1 Example typical pothole photos.....	4
Figure 2.2.1 HSV colour space illustration.....	11
Figure 2.2.2 Light intensity change for various colour schemes.....	11
Figure 2.2.3 Example normalised Gaussian kernel.....	13
Figure 2.2.4 Visual illustration of a Gaussian kernel.....	13
Figure 2.2.5 Sobel operator in mask form.....	16
Figure 2.2.6 Example convex hull algorithm.....	17
Figure 2.3.1 LBP Feature illustration .....	19
Figure 2.3.2 LBP lighting variations example .....	20
Figure 2.3.3 HOG feature descriptor extraction.....	21
Figure 2.4.1 Graphs to show underfitting (a), good fit (b) and overfitting (c) .....	23
Figure 2.4.2 Cascade classifier working .....	24
Figure 2.4.3 SVM hyperplane illustration.....	26
Figure 2.4.4 Sliding window functioning.....	30
Figure 2.4.5 Image pyramid example with scale set to 50%.....	31
Figure 3.4.1 Input data pre-processing .....	42
Figure 3.7.1 Illustration of (a) the pinhole camera model and (b) the corresponding camera plane and image co-ordinate systems .....	50
Figure 3.7.2 Camera calibration .....	52
Figure 3.7.3 Fisheye distortion illustration.....	52
Figure 3.7.4 Fisheye correction measurements.....	53
Figure 3.7.5 Similar triangles illustration .....	55
Figure 3.7.6 Cross ratio point mapping.....	56
Figure 3.7.7 Results of the various approaches .....	60
Figure 3.7.8 Results of the various approaches at 4.93 m to the left of the camera .....	61
Figure 3.7.9 Cross ratio comparisons.....	62
Figure 4.2.1 Road extraction algorithm example.....	67
Figure 4.3.1 Pothole detection algorithm .....	69
Figure 4.4.1 Simple scenario dataset pothole content breakdown. ....	71
Figure 4.4.2 Complex scenario dataset pothole content breakdown. ....	72

Figure 4.4.3 HOG features visualisation .....	73
Figure 5.3.1 Example of the effect of grouping detections .....	83
Figure 5.4.1 Image processing simple scenario precision/recall curves .....	84
Figure 5.4.2 Image processing complex scenario precision/recall curves.....	85
Figure 5.5.1 LBP simple scenario classifier precision/recall curves.....	86
Figure 5.5.2 LBP complex scenario classifier precision/recall curves .....	87
Figure 5.6.1 SVM+HOG simple scenario classifier precision/recall curves .....	88
Figure 5.6.2 SVM+HOG complex scenario classifier precision/recall curves.....	89
Figure 5.7.1 Maximum F1-score classifiers for the simple scenario.....	90
Figure 5.7.2 Maximum F1-score classifiers for the complex scenario .....	90
Figure 5.7.3 Example output of machine learning classifier in simple scenario.....	92
Figure 5.7.4 Example output of machine learning classifier in complex scenario. ....	92
Figure 5.8.1 Simple depth estimation results .....	94
Figure 5.8.2 Complex depth estimation results .....	95

# LIST OF TABLES

Table 2.5.1 CDDMC algorithm results.....	35
Table 2.5.2 Experimental Setup Done by Koch and Brilakis .....	36
Table 2.5.3 Kinect Sensory Key Properties. ....	38
Table 3.6.1 Sample summary.....	46
Table 4.4.1 HOG descriptor parameter values.....	74
Table 5.7.1 Optimal classifier settings and performance based on F1-score.....	91
Table 5.8.1 Detected potholes expressed as a percentage as a function of depth.....	96
Table 5.9.1 Timing performance of the algorithms. ....	98

# ABBREVIATIONS AND ACRONYMS

ANN – Artificial Neural Network

CDDMC – Critical Distress Detection, Measurement and Classification

DFS – Distress Frame Segmentation

fps - Frames Per Second

GPR - Ground Penetrating Radar

GPRS – General Packet Radio Service

GPS - Global Positioning System

HOG – Histogram of Oriented Gradients

HSV – Hue, Saturation, Value

IR – InfraRed

LASER - Light Amplification by the Stimulated Emission of Radiation

LBP – Local Binary Pattern

LCD – Liquid Crystal Display

LOO – Leave-one-out

LRF – Laser Range Finder

LUT – Look Up Table

MLP – Multi-Layer Perceptron

RBF – Radial Basis Function

RGB – Red, Green, Blue

ROC – Receiver Operating Characteristic

SANRAL - South African National Roads Agency Limited

SIFT – Scale-Invariant Feature Transform

SURF – Speeded-Up Robust Features

SVM – Support Vector Machine



# CHAPTER 1

## INTRODUCTION

### 1.1 BACKGROUND

In modern day society, tarred roads have become critical for services and goods to be available to citizens as well as to cater to their transportation needs. The cost of building a tar road is estimated to be up to R 25 million per kilometre [1] and is an expensive necessity. As the demand for food and products to be available across a country increases, the logistical challenges of transporting these items increases, leading to more heavy motor vehicles using the roads. Heavy motor vehicles such as trucks used by freight operators are the biggest culprits in damaging the road surface due to their heavy weight and high axle load. The term axle load is used to describe the force that is exerted on the pavement by a vehicle and is one of the most important considerations when designing a road. A 22 m long heavy motor vehicle can have a load of up to 35 tons [1], which has a larger axle load than a light motor vehicle. One type of damage that is created over a period of time is known as potholes. Two different examples of potholes found in suburban areas are given in Figure 1.1.1.



Figure 1.1.1 Example typical pothole photos. In these photos, several different types of potholes can be seen and in a variety of lighting conditions.

The manner in which a pothole forms is dependent on the type of bituminous pavement surfacing. The volume of traffic and the axle load experienced by the road are example factors that lead to fatiguing of the road surface, resulting in the formation of cracks. These cracks allow

## CHAPTER 1 – INTRODUCTION

---

water to seep through and mix with the asphalt. When a vehicle drives over this area, this water is expelled through the crack with some of the asphalt, and this will slowly create a cavity underneath the crack. Eventually the road surface will collapse into the cavity, resulting in a visible pothole. If regular road maintenance is neglected, the aforementioned cracks are not repaired before they cause substantial damage to the road. Therefore, potholes usually increase in number during and after heavy rainfall seasons [2].

There are sophisticated commercial products available on the market that will scan the road for all types of distress and log the findings for maintenance purposes, such as Roadscanners [3]. These types of vehicles are fitted with a 3D accelerometer, 2D laser scanner, video camera, thermal camera and 3D Ground Penetrating Radar (GPR). All of this adds up financially to a substantial amount and it is therefore normally only large agencies like the South African National Roads Agency Limited (SANRAL) that can afford to purchase these types of vehicles. SANRAL is responsible for the maintenance and planning of the highways in South Africa; therefore, highways tend to be in better condition than municipal roads. Equipment such as a Roadscanner vehicle remains out of reach for municipalities with smaller budgets. This leaves manual pothole detection in towns as the only viable option at this point in time, thereby establishing a need for an inexpensive and accurate pothole detection method that can assist in automating the process.

### 1.2 PURPOSE OF THE PROJECT

The purpose of this project is to investigate the feasibility of automatically detecting potholes with a digital camera. Therefore, a visual approach is required and the images are processed off-line with several different algorithms. The algorithms that are investigated are image processing algorithms, as well as machine learning algorithms. If potholes can be detected, more expansive applications could use this information and incorporate the pothole detection system with other types of technology such as GPRS (General Packet Radio Services) and GPS (Global Positioning System) to log these potholes for maintenance purposes. If the pothole algorithms are found to perform fast enough, it would then also be possible to add a pothole detection feature to driverless vehicles [4, 5].

## CHAPTER 1 – INTRODUCTION

---

### 1.3 RESEARCH OBJECTIVES

The research objectives of this investigation are rooted in the assumption that pothole regions can be distinguished from non-pothole regions in an image. The degree to which this statement is true, will be determined. The investigation will be conducted on a simple scenario and a complex scenario. The simple scenario can be considered to be the ideal case where only open and clear road images are used and therefore the road can be extracted. In the complex scenario, the real-world conditions which include mixed lighting conditions prevented the road from being extracted. Therefore, these images contained foliage and other objects. Three different approaches will be evaluated to analyse the different scenarios, namely an algorithmic image processing approach, and two machine learning approaches.

#### **Research objective 1:**

To investigate the feasibility of detecting potholes in images. Two types of images will be evaluated: a simple scenario and a complex scenario.

#### **Research objective 2:**

To compare the results between an approach using classical image processing algorithms and two suitable machine learning algorithms.

#### **Research objective 3:**

To quantify the effect that potholes have at different distances from the camera on the performance of each of the classifiers.

## CHAPTER 1 – INTRODUCTION

---

### 1.4 SCOPE OF WORK

This study investigates how effectively potholes can be detected by using a visual approach i.e. a camera. This approach required that the camera be mounted to the inside of the windscreen, facing the road. The windscreen of the vehicle was cleaned regularly and therefore the assumption is made that there are no dirty spots on the windscreen, and subsequently, in the images. Due to the physical and visual nature of the problem, only pothole footage during either sunny or cloudy conditions are considered, but not during rainy conditions.

Machine learning algorithms, especially when applied to high-resolution images, can be a very time-consuming task and therefore it was decided to only investigate the LBP (Local Binary Patterns) cascade classifier and the SVM (Support Vector Machine) with HOG (Histogram of Oriented Gradients) classifier. These classifiers are known to have a much shorter training time than that of, for example, a Haar cascade classifier, which is more suited to the time period of the project. Preliminary work where a Haar cascade classifier was trained required seven weeks to complete.

### 1.5 THESIS STRUCTURE

The structure of the rest of this thesis is as follows:

- **Chapter 2: Literature review.** Sections 2.2 – 2.4 of this chapter focusses on explaining the technical detail of the various methods and algorithms used to detect potholes in this thesis. Section 2.5 discusses the various existing works available in the academic community with regards to pothole detection.
- **Chapter 3: Data set collection and preparation.** Sections 3.2 explains how the data was obtained for this project. In Section 3.3, it is discussed that two individual investigations take place in this study namely, a simple scenario and a complex scenario. In the simple scenario, clear open roads are used and the road region within the image is extracted by using the colour of the road before the potholes are detected. In the complex scenario, however, the conditions were varied to include open and mixed lighting conditions. Therefore, the road could not be

## CHAPTER 1 – INTRODUCTION

---

extracted by its colour and the images were cropped to the region in the image where the road is expected to be. This scenario best represents the expected real-world pothole detection problem. The necessary steps to pre-process the data before training/testing could be done is discussed in Section 3.4. Section 3.5 discusses the manner in which the data is annotated, especially for training purposes. The actual number of positive and negatives samples use in the investigation is tabulated in Section 3.6. As it was decided that the specific range in which the pothole detection solution can detect potholes needed to be evaluated, it was necessary to compare different depth estimation techniques in monocular images. This work is presented in Section 3.7.

- **Chapter 4: Development and methodology.** The method for extracting the road surface based on its colour is discussed in Section 4.2. The development of the image processing pothole detection algorithm is discussed in Section 4.3. The methodology that was followed to determine the finer details necessary in the design of the machine learning is discussed in Section 4.4 and 4.5. The machine learning classifiers that are used in this investigation are LBP cascade classifiers and SVM with HOG features classifiers.
- **Chapter 5: Results.** Section 5.2 discusses the performance metric used to compare the various classifiers' performance was selected as precision/recall curves. In Section 5.3 the specific setup used in OpenCV to utilize the machine learning functions is discussed. The remaining sections of Chapter 5 presents the performance results of the classifiers and compares them to one another per scenario being investigated. The classifiers are also evaluated based on the relative distances at which the classifiers could detect potholes (Section 5.8). Lastly, the performance of the classifiers was timed and analysed in Section 5.9.
- **Chapter 6: Conclusion and recommendations.** In the last chapter, the final conclusions that were found during the course of the project based on the results presented in Chapter 5 is given. Lastly, recommendations for improvement and future work are also given and explained.

# CHAPTER 2

## LITERATURE REVIEW

### 2.1 INTRODUCTION

This chapter reviews the relevant literature with respect to the image processing algorithms and machine learning algorithms applied in this thesis. The image processing algorithms/techniques discussed are usually only applicable to single channel (for example, the red colour matrix in an RGB image). If it is necessary to apply these algorithms to a full RGB image, the image must first be split into its three respective channels, the algorithm applied to each channel respectively, and the results are merged together to create the new RGB image.

The algorithms and techniques are divided and discussed in different sections. Section 2.2 contains the necessary theory to understand the image processing methodology used in Chapter 4. This includes information regarding the colour space, edge detection and noise removal techniques used. The operation of the contour detection and convex hull algorithm is also discussed in Section 2.2. In Section 2.3, the various types of features used are explained. After Section 2.3, the specific machine learning algorithms are explained in Section 2.4. The two different types of machine learning algorithms that will be used in this project is a cascade classifier and an SVM (Support Vector Machine). The theory with respect to the application of the machine learning algorithms such as the sliding window and image pyramid techniques, is also discussed. Due to the fact that features can also be used in the absence of machine learning to detect objects, these sections are separated. In Section 2.5, the papers that are directly involved with pothole detection are discussed. From the papers it was found that various technologies can be applied to pothole detection such as accelerometers to measure the vibrations of the vehicle and infra-red cameras that obtain depth maps of an area. Lastly, a conclusion regarding all of the theory is presented in Section 2.6.

Note that all of the algorithms were implemented by using OpenCV 2.4.8 [6] which is an open-source computer vision library. For practical purposes, it is also important to note that in OpenCV, the top left pixel is the reference point and is therefore  $(x,y) = (0,0)$ . If a pixel's colour

## CHAPTER 2 – LITERATURE REVIEW

---

is black, the value in OpenCV is 0 and if a pixel is white, its value is 255. Where applicable in this chapter, as well as the rest of the thesis, this information should be kept in mind.

### 2.2 IMAGE PROCESSING

This section contains the relevant image processing theory necessary for the comprehension of the image processing algorithm developed in Chapter 4.

#### 2.2.1 COLOUR SPACES

The term colour space refers to the collective term to describe the particular mathematical representation of the different colours as well as their saturation and contrast [7]. The RGB (Red, Green, Blue) colour space consists of a red, green and blue channel and is very commonly used to represent images in a variety of applications, such as LCD (Liquid Crystal Display) screens [8]. A colour channel can be thought of as a matrix containing the exact pixel values for each of the primary components of the chosen colour model [9]. In general, a colour space consists of three different channels. The RGB colour space is not appropriate when processing images, because it is difficult to discriminate between the different colours [10]. A much more suitable colour space is the HSV (Hue, Saturation, Value) colour space as it separates the chroma (colour information) from the luma (intensity information) in the channels which in turn can simplify the task of detecting objects within an image [11]. An example illustration of this colour space is shown in Figure 2.2.1. From the figure, it is clear that the colour space uses a conical shape to represent the various colours. The hue is expressed on a circular scale and is therefore either represented in degrees or radians. The saturation is expressed as the radius of the cone at a particular height from the vertex of the cone, while the value channel is expressed in terms of the height of the cone. HSV performs well in conditions where there is a high variety in light intensity and can be observed in Figure 2.2.2 [12]. In the figure, the light intensity was varied and it was found that by using the HSV colour space the colours of the image could be more accurately extracted than in the RGB colour space.



## CHAPTER 2 – LITERATURE REVIEW

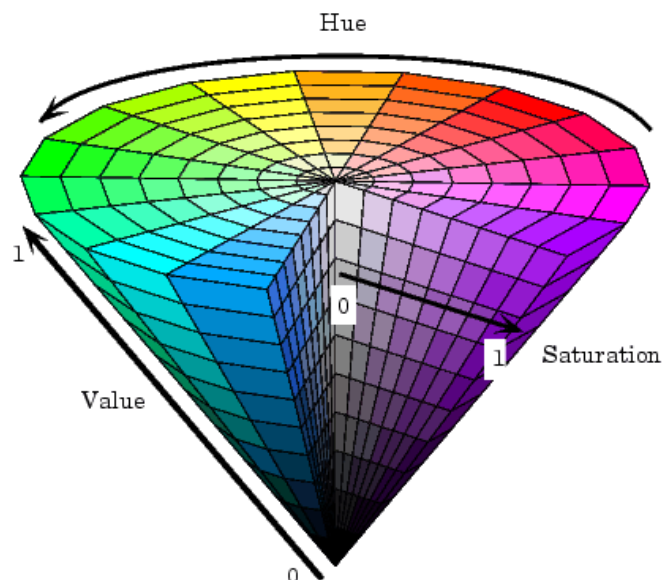


Figure 2.2.1 HSV colour space illustration [13]. The colour space has a conical shape which separates the colours more evenly than the RGB colour space. Hue is measured on a circular scale and is indicated in either radians or degrees. The saturation value is measured with respect to the centre of the cone and is akin to its radius at a particular height from the vertex of the cone. The light intensity is represented by the value and is measured from the tip of the cone to the particular colour.

The HSV colour space does, however, not perform well if the colour of the object that needs to be detected varies over a wide range. For example, HSV is not the best suited colour space to detect objects that are orange and have a wide range of saturation for the object.

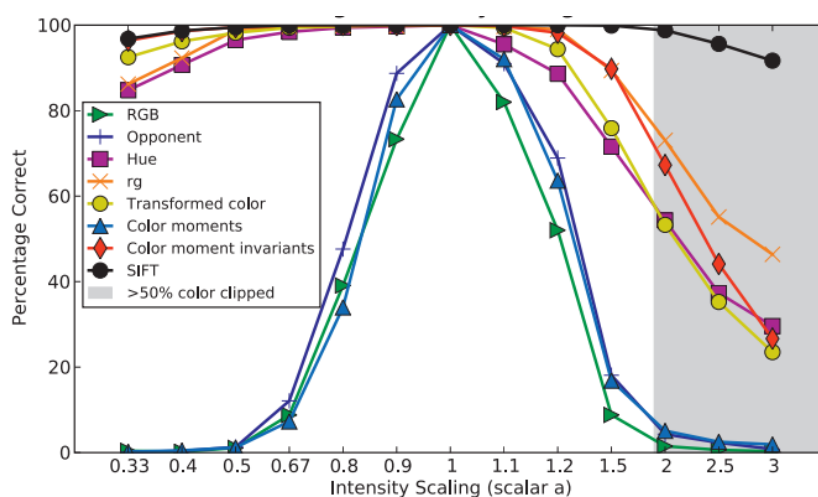


Figure 2.2.2 Light intensity change for various colour schemes [12]. A variety of colour spaces are weighed against each other over change in intensity. The only colour spaces implemented in OpenCV are the RGB and hue colour spaces. From the figure, the hue channel remains much more correct for a wider range than for the RGB colour space.



## CHAPTER 2 – LITERATURE REVIEW

---

However, in this project, the colour information of the pothole is not used to detect the pothole, therefore, the HSV spectrum is a suitable choice.

### 2.2.2 NOISE REMOVAL TECHNIQUES IN IMAGES

This section discusses the different noise removal techniques used in this thesis namely: Gaussian blur and morphological operations such as erosion and dilation.

#### 2.2.2.1 GAUSSIAN BLUR

Blurring is a method for smoothing images to remove noise [14]. There are many different types of kernels that can be used to smooth an image such as average (also referred to as a box filter), median and Gaussian. Each type of blurring function uses a kernel to scan the image and by performing kernel-specific calculations, a smoothed output image is produced. These kernels are usually of the form  $A \times A$  pixels where  $A$  is an odd integer number of pixels. The average kernel calculates the average value of the pixels in the kernel window, normalises the average value and outputs this value to the  $x$ - and  $y$ -coordinate of the centre pixel of the kernel at the output image. The kernels are normalised to ensure that the output image does not increase in brightness and keeps the colour information accurate [15]. Similarly, the median kernel will use the median value of the kernel to determine the output. A Gaussian kernel uses the Gaussian bell curve to perform its calculations. The Gaussian operator was used in this project as it is required by the Canny edge detector (discussed in the next section).

The Gaussian operator given a two-dimensional space (such as the  $x$ - and  $y$ -coordinates of a Cartesian plane) is given in equation (2.2-1). The standard Gaussian operator includes a  $\mu$ , and refers to the position of the centre of the peak, however, in image processing, this parameter does not feature in this equation as the Gaussian bell curve is centred about the origin and is therefore zero [16].

$$g(x,y) = ce^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (2.2-1)$$

## CHAPTER 2 – LITERATURE REVIEW

An example of a normalised 3 x 3 Gaussian kernel is shown in Figure 2.2.3 [16].

$$\frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Figure 2.2.3 Example normalised Gaussian kernel. The kernel has a size of 3 x 3 pixels and is an example approximation of the Gaussian kernel. Note that to normalise the output of the kernel, the kernel is divided by the sum of the internal components which is 16 for this particular kernel.

The visual representation of a large Gaussian kernel is given in Figure 2.2.4. The smaller picture in the bottom right corner illustrates what the Gaussian kernel intensity values are that will be applied to the pixel regions included in the kernel. The higher the value of the intensity, the lighter the pixel becomes and if the value is smaller, the pixel value becomes darker.

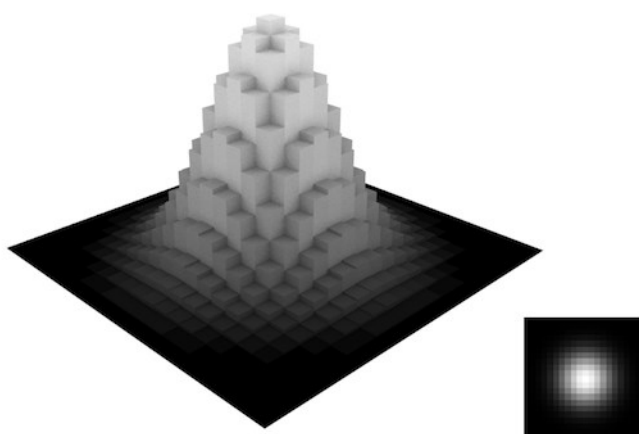


Figure 2.2.4 Visual illustration of a Gaussian kernel [17]. The centre figure illustrates the bell curve that is characteristic of the Gaussian kernel. As the pixel values within an image is discrete, the curve is also discrete. In the bottom right corner, an example of the kernel as it is applied to an image is illustrated.

### 2.2.2.2 EROSION AND DILATION

There are a variety of morphological operations such as erosion and dilation [18]. The erosion and dilation operations can be used in to remove residual noise from an image that is typically created when edge detection has been implemented [19]. In this project, both erosion and

## CHAPTER 2 – LITERATURE REVIEW

---

dilation are only applied to binary (black-and-white) images and are therefore discussed in terms of how these two operations function on binary images.

Practically, the erosion binary operation increases the number of black pixels in a binary image while the dilation binary operation increases the number of white pixels. In both operations, there is a kernel that is convoluted with the input image to produce an output [14]. In OpenCV, this kernel has a square shape of size  $A \times A$  where  $A$  is any odd integer above 1.

In the case of the dilation kernel, it effectively scans the pixels of an input image and if there are any white pixels (maximum pixel value) present in the “window” of the kernel, it sets the output pixel at the coordinates of the centre pixel of the kernel, to the maximum pixel value. Similarly, the erosion kernel determines the minimum pixel value. This is illustrated by equation (2.2-2) [14]. In the equation  $src$  refers to the input source pixels.

$$\begin{aligned} \text{erode}(x,y) &= \min_{(x^k,y^k) \in \text{kernel}} \text{src}(x + x^k, y + y^k) \\ \text{dilate}(x,y) &= \max_{(x^k,y^k) \in \text{kernel}} \text{src}(x + x^k, y + y^k) \end{aligned} \quad (2.2-2)$$

### 2.2.3 CANNY EDGE DETECTION

An edge represents a sharp gradient change between neighbouring pixels. Accurate edge detection in images are important for object detection because if the edges of the objects within an image can be extracted; the shape, area and perimeter of an object can be determined which would aid in the successful detection of a particular object [20]. Once the edges have been detected, it is possible to, for example, detect the contours in an image. This process is referred to as edge tracing and it follows either a clockwise or anti-clockwise approach. By picking a particular edge pixel and determining its next clockwise neighbour, a linked list of the data can be created which will describe a continuous contour. Edge detection is also seen as a form of image segmentation as it can be used to segment the image into different regions [19, 20].

A variety of edge detection algorithms are available such as the Prewitt [21], Roberts cross detector [22], Sobel edge detector [23], Scharr [24] and Canny [25].

The Canny edge detector was developed with the following three criteria [26]:

## CHAPTER 2 – LITERATURE REVIEW

---

1. Detection: The edge detector must not be sensitive to noise and must detect the maximum amount of edges in an image that is possible.
2. Localization: The maximum point of the edge calculation should indicate where the true edge location is.
3. Minimal response: There must be no more than one detected edge per edge present in the image.

Due to the above mentioned criteria, the Canny edge detector was selected for use in this thesis.

There are five steps in the Canny edge detector algorithm namely: smoothing, finding gradients, non-maximum suppression, double thresholding and edge tracking by hysteresis [26].

As was mentioned in the previous section, the standard smoothing method used in conjunction with the Canny edge detector is a Gaussian blur/filter [26, 27].

Then, the edges within images can be determined by determining the rate of change of the gradient with respect to the intensities in an image. A method for calculating the gradient of a function  $f(x,y)$  is to determine its derivative which leads to:

$$\nabla f(x,y) = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = (\mathbf{g}_x, \mathbf{g}_y) \quad (2.2-3)$$

where the partial derivative of the x- and y-axis pixels are denoted by  $\mathbf{g}_x$  and  $\mathbf{g}_y$ . A larger derivative would indicate a sharper edge while a smaller derivative indicates a less sharp edge. A derivative equal to zero indicates no gradient change and therefore no edge is present. The partial derivatives  $\mathbf{g}_x$  and  $\mathbf{g}_y$  are vectors and the magnitude and direction of the gradient can be determined simply by

$$\|\nabla f(x,y)\| = \sqrt{\mathbf{g}_x^2 + \mathbf{g}_y^2} \quad (2.2-4)$$

and

$$\text{Direction} = \arctan\left(\frac{\mathbf{g}_y}{\mathbf{g}_x}\right) \quad (2.2-5)$$

## CHAPTER 2 – LITERATURE REVIEW

---

Each of the gradients are calculated by using the Sobel-operator [23] which is given in mask form in Figure 2.2.5. The horizontal gradient operator on the left hand side of the image, detects horizontal edges from left to right while the vertical gradient operator on the right hand side of the image detects vertical edges from the bottom to the top.

-1	0	1	1	2	1
-2	0	2	0	0	0
-1	0	1	-1	-2	-1
Horizontal gradient operator			Vertical gradient operator		

Figure 2.2.5 Sobel operator in mask form. The gradient is determined separately in the x- and y-direction by using two different operators. The horizontal gradient operator is found on the left hand side and the vertical gradient operator is on the right hand side.

Due to the previously mentioned step, the output of the Canny edge detector can be interpreted as a gradient image. In the gradient image, pixels that are black are pixels that have a low gradient while the white pixels represent pixels with a higher gradient. In this manner, the edges are represented by gradients. The next steps simply add additional robustness to the algorithm according to the previously stated criteria.

The next step in the Canny edge detector, is non-maximum suppression which effectively narrows the detected edges. For each detected edge pixel from the previous steps, the maximum gradient magnitude is determined by evaluating its direct neighbours along the direction of the gradient at that edge pixel and only the pixel with the maximum gradient magnitude is selected. This also sharpens the detected edges output image.

A thresholding algorithm which has two thresholds, one low and one high, is applied. If a pixel gradient is below the lowest threshold, it is discarded and a pixel above the highest threshold is deemed a strong edge. Pixel gradients that fall in between the high and low threshold are deemed weak edges.

The last step performs a hysteresis operation. A hysteresis operation has two threshold values and if an edge value is below the lower threshold, it will reject the edge. Similarly, if an edge value is above the upper threshold, it will accept the edge. In this manner, the hysteresis function

## CHAPTER 2 – LITERATURE REVIEW

---

traces out the contour of the edge in such a manner that it fits the edges to a connected pixel line and prevents disconnected lines on a single contour.

### 2.2.4 CONTOUR DETECTION

A contour in an image is the outline/boundary of an object in the image [14] and as stated in Section 2.1, contours in image processing comprise linked lists of neighbouring pixels. By using, for example, an edge detection technique, the output binary image can be fed to a contour detection algorithm. The algorithm will create closed loop contours as contour objects in the software which will then automatically determine the exact location, length and area of the contour.

Naturally, it is possible to have multiple contours in an image. For these cases, OpenCV builds hierarchical structures where each element in the structure refers to a contour. Dependant on the type of structure that the user specifies, it is possible to extract only the smaller contours that are found within a larger contour.

### 2.2.5 CONVEX HULL ALGORITHM

A convex hull algorithm in image processing constructs a convex polygon by using all of the furthest pixel points of a particular contour. An example application that demonstrates this algorithm is shown in Figure 2.2.6.

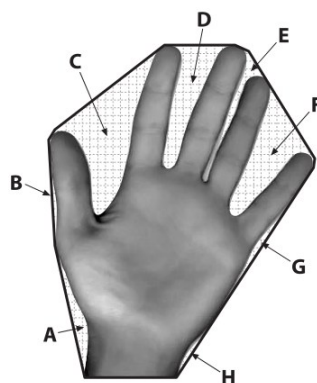


Figure 2.2.6 Example convex hull algorithm [14]. A convex polygon is fit to the hand in the figure. Points A-H indicate the valleys that are included in the convex polygon that would not be included in a standard contour detection algorithm.

## CHAPTER 2 – LITERATURE REVIEW

---

Each of the indentations of the hand are marked (A-H) and indicate the gaps in the figure with respect to the outlying points. These gaps are included in the convex hull because the convex hull algorithm will connect each of the furthest points together with straight lines and would encompass the entire hand.

### 2.3 FEATURE TYPES

Features, in terms of images, are seen as points of interest and for a particular set of features to be useful in object detection, it is necessary that the object features need to be consistent across all of the instances of the object in all of the images [28].

Careful consideration must be given when choosing a particular feature type as to whether it is applicable to the object that is desired to be detected. In the case of pothole detection, the pothole itself is usually firstly characterized by a dark/black region (sources) and surrounded by lighter regions. Therefore, features that specialize in gradient changes a.k.a. edge detection are a good option for this particular object.

There are many different types of features that can be investigated. Some, such as SIFT (Scale-Invariant Feature Transform) [29] and SURF (Speeded-Up Robust Features) [30], are bounded by a patent and therefore need to be paid for especially if a commercial product is desired [31]. Therefore, this thesis is limited to features that are free and available within OpenCV.

In OpenCV, there are Haar [32], LBP (Linear Binary Pattern) [33] and HOG (Histogram of Oriented Gradients) features [34, 35]. Both the Haar features and the LBP features are known to not be rotation invariant [36, 37]. Therefore, these features are not designed for objects that have a high variety in their rotation in the image. HOG features were specifically developed to detect humans in images, with the emphasis on being able to detect humans in a variety of different poses and in different situations/backgrounds. HOG features, therefore, specialise in detecting objects with a high variety in shape which implies that these features could be a good set of features to consider when detecting potholes as potholes vary greatly in shape.

Initial tests done in this works indicated that the process of training a cascade classifier using Haar features on high-resolution images required upwards of six weeks. Therefore, due to time constraints, these features were excluded from the investigation.

## CHAPTER 2 – LITERATURE REVIEW

### 2.3.1 LOCAL BINARY PATTERNS

In [33], LBP (Local Binary Patterns) features were first presented. LBP features are texture operators functioning in a two-dimensional space and can therefore be used to extract texture features in images. The textures in an image can be described as particular patterns present in an image such as a checkerboard pattern or the ridges found in fingerprints. LBP features are determined by thresholding the 8 nearest neighbours around a pixel by the centre pixel's value. Any value equal to or below the centre pixel is deemed a zero and if the value is higher than the value of the centre pixel, the new value becomes 1. An example of this thresholding transformation of pixels is demonstrated in Figure 2.3.1 by using a 3 x 3 window.

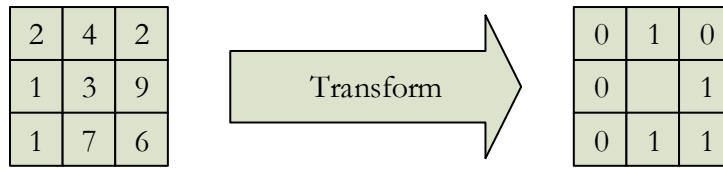


Figure 2.3.1 LBP Feature illustration. On the left hand side, there is a sample of 3 x 3 pixels in a window. By using the centre pixel as the threshold, the transformed output window on the right hand side is generated.

The binary interpretation of the output (starting in a clockwise motion from the top left pixel) yields an 8-bit code of 01011100. The 8-bit code is then used and the number of transitions from zero to one is counted as patterns. A histogram is then formed based on the patterns as opposed to the values of the 8-bit code which reduces the size of the total LBP feature. An input sample image is then scanned with the window and a total histogram representation of the sample is determined by summing each window's histogram. LBP features can also be represented mathematically as an operator such as in equation (2.3-1) [38]. As the output of the operator is interpreted as binary code, the function in the equation is multiplied by  $2^p$  which converts the binary code to decimal.

$$LBP_{p,R}(x_c, y_c) = \sum_{p=0}^{P-1} \text{sign}(g_p - g_c) 2^p \quad (2.3-1)$$



## CHAPTER 2 – LITERATURE REVIEW

---

Where:

$(x_c, y_c)$  = centre pixel

$\text{sign} = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$

$P$  = number of pixels

$R$  = radius of the pixels ( $P$ ) on a circle from the centre

$g_p$  = gray values of  $P$  pixels equally spaced on radius  $R$

$g_c$  = gray value of centre pixel

LBP features are considered in this thesis due to their inherent immunity to a vast variety in lighting conditions as is demonstrated in Figure 2.3.2. The contrast of the images in the top of this figure are varied widely and their respective LBP features' visual representation is shown at the bottom of the figure. The LBP features' visual representations hardly change as the contrast varies which makes it a suitable feature to extract in situations where lighting variations are inevitable. Potholes occur outside in an uncontrolled environment and therefore, images containing potholes have a high variety in their contrast. On occasion, glare is also present in the images of the potholes, which also impact the image's contrast which produces an image similar in appearance to the image second from the left in top part of Figure 2.3.2. Hence, LBP features could be viable features in pothole detection.

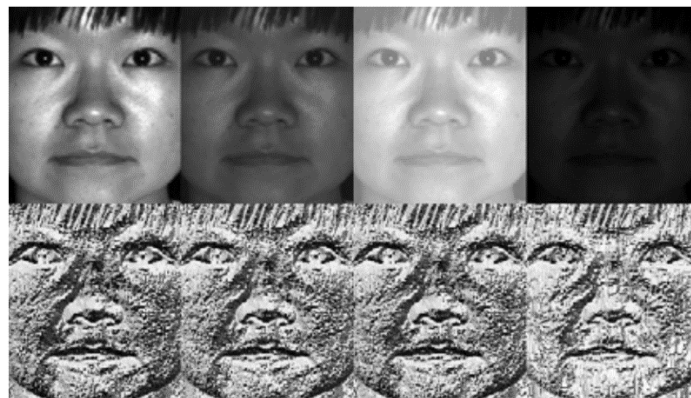


Figure 2.3.2 LBP lighting variations example [39]. A sample face is taken and the contrast is manually varied over a wide spectrum as can be seen in the top images of the figure. The corresponding LBP image is given in the bottom row.

## CHAPTER 2 – LITERATURE REVIEW

### 2.3.2 HISTOGRAM OF ORIENTED GRADIENTS

A basic diagram illustrating the extraction of HOG feature descriptors is shown in Figure 2.3.3. The first step is optional and comprises the normalisation of the gamma and colour of the image, and is introduced to compensate for a variation in lighting conditions. Gamma is a measure for the relationship between a pixel's value and its luminance [40]. The normalisation is achieved by either determining the square root or log of the image intensity per channel [41].

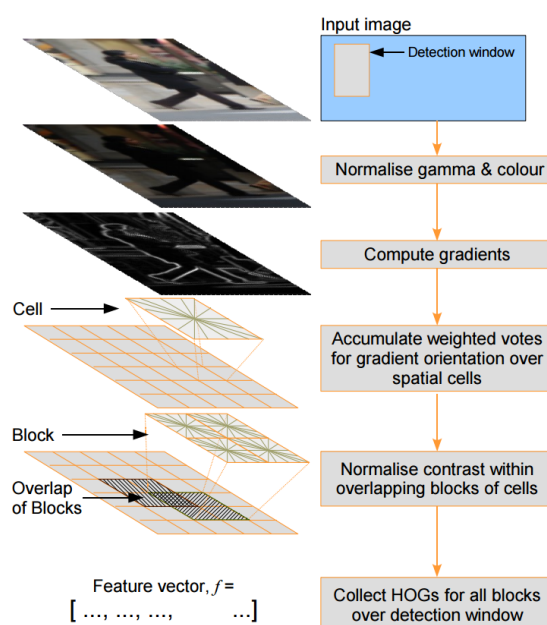


Figure 2.3.3 HOG feature descriptor extraction [35]. The figure illustrates the steps required to determine the feature vector for detection. The first step is optional and will normalise the gamma of the image. Thereafter the gradients are computed per pixel. The gradients have both a magnitude and direction which can be used to obtain a histogram of orientations by sorting the magnitudes into bins based on their direction. Several cells are grouped together to form a block over which normalisation will take place. Lastly, the HOG of each window is placed in a feature vector and can be used for detection purposes.

In the second step the gradients within the image is computed (see Section 2.2.3). To determine the particular magnitude and direction of the gradient change, equations (2.2-4) and (2.2-5) are applied. From the third step onwards, to compute the HOG feature descriptor, it is necessary to divide the image into cells and blocks where the dimensions (width and height) of these components are specific to the object being detected. In the third step, a histogram of orientation is created per cell. The histogram is created by firstly deciding how many bins there should be in the histogram and what the maximum allowed orientation of the gradients are. For

## CHAPTER 2 – LITERATURE REVIEW

---

example, if there are 9 bins, and the maximum allowed orientation of the vectors are chosen to be 180 degrees, the bins will be divided into 20 degree intervals (0 - 20 degrees, 20 - 40 degrees ... 160 - 180 degrees). Then the magnitude of each of the vectors are added to the orientation bin to which the vector belongs to form a histogram. This process is done per cell and the number of bins and the maximum allowed orientation are kept constant.

The algorithm is made more robust in terms of lighting variations by combining several cells into a block and normalising over the block. Each block is then placed into a feature vector which can then be used as an input to an algorithm (such as a machine learning algorithm) to detect objects. The feature vector is also referred to as the HOG descriptor and its length can be calculated as in equation (2.3-1) [42].

$$\text{HOG descriptor length} = \# \text{Blocks} * \# \text{CellsPerBlock} * \# \text{BinsPerCell} \quad (2.3-2)$$

## 2.4 MACHINE LEARNING

### 2.4.1 INTRODUCTION

The point of applying machine learning to a problem is to develop a function that is able to take the input training data and create a model that can generalise well enough with respect to the data that it can provide accurate predictions on new data. If there is a wide variety in the data per class and not many similarities, it becomes difficult for the classifier to build a model that generalises satisfactorily [43]. Figure 2.4.1 illustrates the various types of fitting that can occur when training a two-class classifier. Figure 2.4.1.a and Figure 2.4.1.c both indicate model behaviour that is unwanted and is referred to as underfitting and overfitting, respectively. Note that when underfitting occurs, the model underestimates the data and more data is misclassified to the wrong class. When a model overfits, it fits the training data precisely but usually underperforms on test data as the model does not generalise enough. The most desired fitting for the data is shown in Figure 2.4.1.b which shows a good generalisation of the data without the model being too complex. There are a large number of options available when choosing a machine learning algorithm such as cascade classifiers [14, 32], SVMs (Support Vector Machines) [44] and ANNs (Artificial Neural Networks) that consist of MLP (Multi-Layer Perceptrons) [45].

## CHAPTER 2 – LITERATURE REVIEW

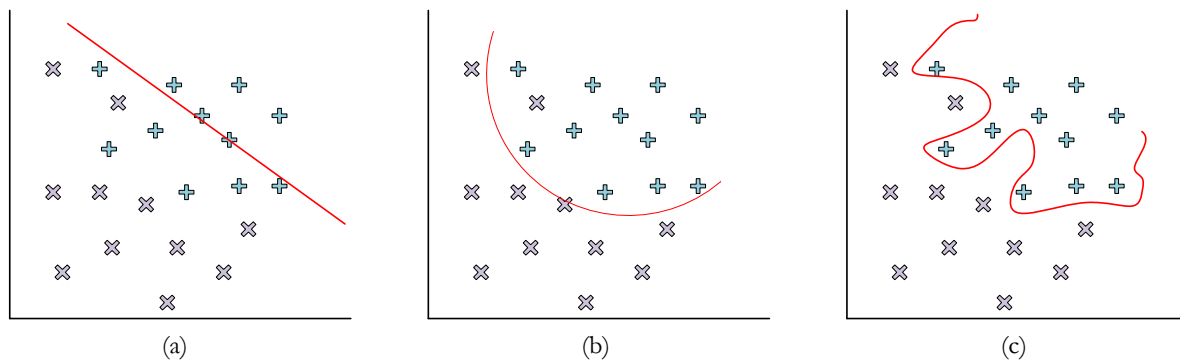


Figure 2.4.1 Graphs to show underfitting (a), good fit (b) and overfitting (c). The graphs indicate the result when a two-class classifier is required. With underfitting, it is clear that a large portion of the data of one of the classes (blue pluses)

Two different classifiers were selected in this thesis namely, a cascade classifier and a SVM (Support Vector Machine) which are discussed in this section. Thereafter, relevant important theory w.r.t. the classifiers as applied in this thesis is presented.

### 2.4.2 CASCADE CLASSIFIER

#### 2.4.2.1 STANDARD CASCADE CLASSIFIER OPERATION

Cascade classifiers use several stages to classify data [14, 32]. The general structure of a cascade classifier is given in Figure 2.4.2. There are several stages directly connected in this structure. An input sample that needs to be classified is presented to the first stage which will either conclude that it is a negative sample and discard it or it will conclude that the sample *might* be a positive sample and, therefore, further testing of the sample is needed by sending the sample to the more advanced stages further down the structure in the cascade. The particular feature type (such as LBP) used during the training phase of the classifier, is also used to perform mathematical operations on the input sample. During training, different mathematical operations were executed for each stage of the classifier and yielded in certain thresholds for each stage. Therefore, a positive samples needs to fall within each of the trained thresholds for each stage in order to be classified as a positive sample. If at any point, any stage concludes that the input sample is a negative sample (by falling outside the particular threshold levels for a stage), it is labelled as a negative sample and the sample is not investigated further.

In unbalanced systems where there are more negative than positive samples, the probability of a sample being negative is much higher than the occurrence of a positive sample. For these

---

## CHAPTER 2 – LITERATURE REVIEW

---

types of systems, it is advantageous to use cascade classifiers as it is rarely necessary to perform the necessary calculations required for all of the stages for each sample which leads to a faster run-time.

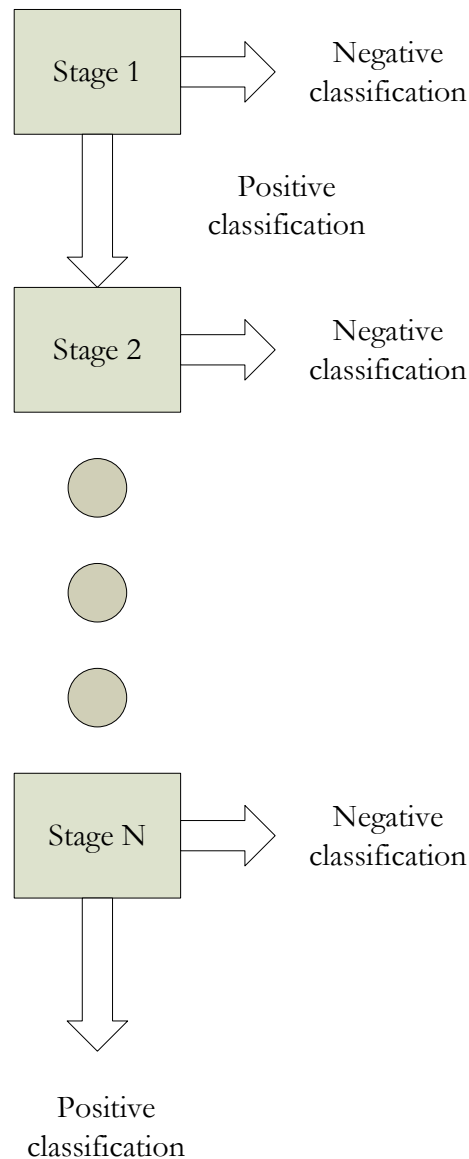


Figure 2.4.2 Cascade classifier working. Several stages are connected together where the positive classification of the previous stage is fed into the input of the next stage. If at any point, a stage classifies the input sample as negative, the sample is labelled as negative and the process exits. A true positive sample must be classified as positive by each of the stages in order to be classified correctly.

In [32], each of the stages consists of a single weak classifier. A weak classifier is a classifier that performs poorly on its own but is slightly better than randomly guessing. Weak classifiers are

---

## CHAPTER 2 – LITERATURE REVIEW

---

by definition not good enough to be used on their own, however, by combining them in a cascade manner, it is possible to create a very good classifier, which is discussed in the next section namely, boosting.

### 2.4.2.2 BOOSTING

The term boosting was first postulated in [46] which posed the question, whether using a number of weak learners/classifiers together could result in a stronger classifier with better accuracy. Boosting can be applied to cascade classifiers to improve their performance [32]. With boosting there are a series of iterations of the algorithm that takes place [47]. These iterations are used to determine the particular weight of a sample. It is required that all of the samples are labelled.

The boosting algorithm, was first introduced in [48] and is explained as follows: for all of the samples, the weight per sample start off as the same value. A decision stump then selects a sample. The feature value at that particular sample is then selected as the threshold value and, consequently, the weak learner classifies all of the samples below this threshold as positive and all of the samples above this as negative. Then, an update of the samples weights take place and the misclassified samples' weights are increased. A new sample and threshold value is selected, based on the weights of all of the samples. The process continues for several iterations in a similar fashion. Lastly, all of the weak classifiers with their respective thresholds are aggregated together to form one classifier. Therefore, boosting effectively is a method for selecting the appropriate weak learner per iteration. This specific meta-algorithm that evaluates the samples based on their weights and updates the misclassified samples is known as AdaBoost [49].

### 2.4.3 SVM CLASSIFIER

SVMs are based on the theory of statistical learning and were first developed in [44, 50]. If it is assumed that the data is linearly separable, a linear support vector machine for a two-class problem can be used.

## CHAPTER 2 – LITERATURE REVIEW

To explain the basic concept of a SVM, the case of the linear two-class SVM is considered in Figure 2.4.3. Given a plane such as the one in the figure and the example sample set in the plane, there exists a straight line that is equally spaced between the positive and negative samples which can be represented by the red dashed line. This line is known as either the decision boundary or the optimal hyperplane. The purpose of the SVM is to find the optimum placement of this hyperplane such that the margin (distance between the two red solid lines) is a maximum. The red line to the top of the optimal hyperplane is known as the positive boundary/hyperplane and is measured from the decision boundary to the closest positive samples that falls on the positive boundary. Similarly, there exists a negative boundary/hyperplane to the bottom of the decision boundary. The samples that fall on either the positive or negative boundaries, are known as support vectors. These support vectors are the only data points relevant to the calculation of the maximum hyperplane margin.

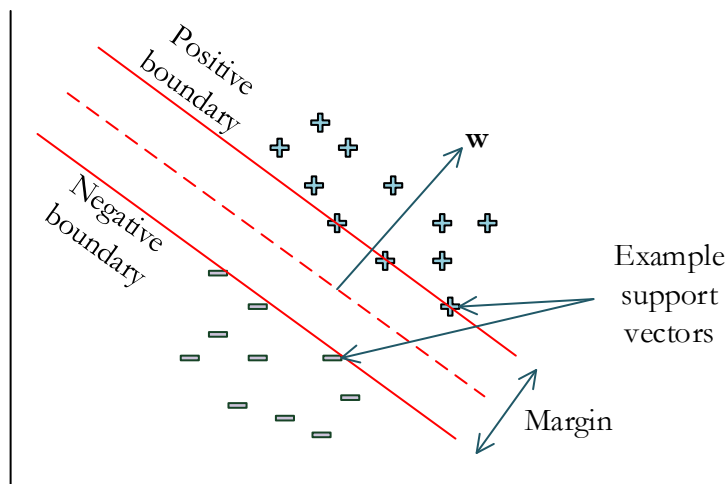


Figure 2.4.3 SVM hyperplane illustration. The feature set contains separable positive and negative samples. The decision boundary is represented with the red dashed line and a maximum margin (distance between the two red lines) is desired. Note that the decision boundary is equidistant from the positive and negative boundaries.

The derivation of the SVM is as follows: first, consider training data  $\mathbf{x}_i$ , where  $\mathbf{x}_i \in \mathbb{R}^m$  and has a label,  $y_i \in \{-1, +1\}$ . In the figure,  $m = 2$ . Therefore the data are real elements in a 2-dimensional space. A positive sample is labelled as +1 and a negative sample is labelled as -1. The optimal hyperplane is expressed as:

## CHAPTER 2 – LITERATURE REVIEW

---

$$\mathbf{w} \cdot \mathbf{u} + b = 0 \quad (2.4-1)$$

where  $b$  is an unknown bias constant and  $\mathbf{w}$  is a weight vector that is perpendicular to the decision boundary.

The decision rule for a positive classification is equal to

$$\mathbf{w} \cdot \mathbf{u} + b \geq 0 \quad (2.4-2)$$

and

$$\mathbf{w} \cdot \mathbf{u} + b < 0 \quad (2.4-3)$$

for a negative classification. To solve the problem, we insert a positive and negative sample into the equation which leads to the following constraints:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_+ + b &\geq +1 \text{ when } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_- + b &\leq -1 \text{ when } y_i = -1 \end{aligned} \quad (2.4-4)$$

Combining these equations with their respective labels leads to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_+ + b) \geq 1 \quad (2.4-5)$$

and

$$y_i(\mathbf{w} \cdot \mathbf{x}_- + b) \geq 1, \quad (2.4-6)$$

and finally a more general equation of

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad (2.4-7)$$

Now, to optimise the spacing of the decision boundaries, it is desired to have boundaries that are as wide as possible. The width of the boundaries can be determined by projecting the normalised vector  $\mathbf{w}$ , onto the difference vector.



## CHAPTER 2 – LITERATURE REVIEW

---

$$\text{width} = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \quad (2.4-8)$$

From manipulating equations (2.4-4) and substituting the results into equation (2.4-8), the width of the margin is found to be  $2/\|\mathbf{w}\|$ . To determine the maximum width of the margin is mathematically equivalent to finding the minimum of the reciprocal of the margin. By mathematically manipulating and simplifying the maximum margin, equation (2.4-9) is found.

$$\max \frac{1}{\|\mathbf{w}\|} = \min \frac{1}{2} \|\mathbf{w}\|^2 \quad (2.4-9)$$

The minimization is accomplished by applying Lagrangian multipliers ( $\alpha_i$ ) as the decision vector  $\mathbf{w}$  is the linear sum of some of the samples (the others equate to zero) [51]. Therefore, to solve the optimisation problem equation (2.4-10) is used.

$$L = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (2.4-10)$$

Minimising equation (2.4-10) over  $\mathbf{w}$  by finding the partial Lagrangian derivatives and setting them to zero yields:

$$\mathbf{w} = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i \quad (2.4-11)$$

Similarly, by minimizing the same equation over  $b$  yields:

$$\sum_{i=1}^l \alpha_i y_i = 0 \quad (2.4-12)$$

Substituting these equations back into the Lagrangian equation gives:

$$L = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (2.4-13)$$

## CHAPTER 2 – LITERATURE REVIEW

---

Finding the optimal parameter  $\alpha^o$  by applying the constraint of equation (2.4-12) and noting that all values for  $\alpha \geq 0$  will lead to the solution of the optimal hyperplane where  $b_o$  is the value of  $b$  necessary for the optimal hyperplane:

$$\sum_{i=1}^l \alpha_i^o y_i \mathbf{x}_i + b_o = 0 \quad (2.4-14)$$

The above process can be modified and repeated for the case where the data is not linearly separable and is known as soft margin classification [52].

From the SVM derivations on the previous pages, it was shown that the maximisation depends on only the vector dot product of the data vectors. The dot product between any two vectors has the property of  $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$  where  $K(\mathbf{x}_i, \mathbf{x}_j)$  is a kernel function. A kernel function can be used to perform the transformation in the vector space that separates the positive and negative samples/vectors. Example kernels [52] that are used with SVMs are:

- Linear:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial:  $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$
- RBF:  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$
- Sigmoid:  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$

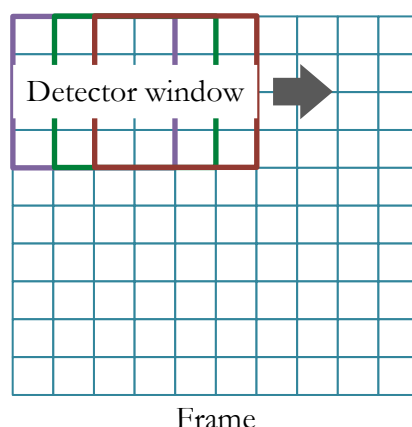
The choice of the kernel is a crucial factor in the performance of the SVM [53] and therefore careful consideration of the input data is necessary in order to select the most appropriate kernel.

### 2.4.4 SLIDING WINDOW AND IMAGE SCALING

A common approach to detect an object in an image is to apply a sliding window (usually much smaller than the original input image) to an image and scan the image for instances of the object. Another approach would be to detect any strong feature points in an image and try to match them to a trained model of the object. The latter is usually the case with SIFT and SURF feature extraction to perform image matching. Figure 2.4.4 illustrates the sliding window process where a square image of size 10 x 10 pixels is scanned with a 4 x 4 pixel sliding window (a.k.a.

## CHAPTER 2 – LITERATURE REVIEW

detector window). The detector window starts in the top left corner and scan towards the right until it reaches the end of the image. It will then move one row down, and the process is the repeated. For each instance of the window in the image, the chosen features within the image is calculated and fed to the classifier for classification.



**Figure 2.4.4 Sliding window functioning.** An example input image of size 10 x 10 pixels is scanned with a detector window of 4 x 4 pixels. For each position of the detector window, features are calculated and used for detection purposes.

This process can only be used in instances where the object that needs to be detected is always of the same size. If the object size differs, it is necessary to either scale the image down by a factor repeatedly or to scale up the detector window by a factor repeatedly. This concept is demonstrated in Figure 2.4.5 which shows an example of how an image is repeatedly downsampled. In the figure the original image is scaled by 50% several times to produce the image pyramid. A sliding window approach will be applied to each of the scaled down images to detect the specific object in question. By using this approach, a scale-invariant classifier can be created to certain extent. This concept will be further discussed and demonstrated in Chapter 5.

There are some drawbacks to using a windowing function [55]. The first being that because the entire image is scanned for an object, any contextual information such as the usual region where the object is found is lost. In the case of potholes, for example, it is highly unlikely that a pothole will occur in the sky region of the image. The windowing function, will however, scan the entire image which increases the chance of detecting false positives. The second drawback of using a windowing function is the aspect ratio of the window is fixed which implies that the detected object must have the same fixed aspect ratio, otherwise, it cannot be detected. Potholes are not

## CHAPTER 2 – LITERATURE REVIEW

fixed objects and have a high variation in appearance and size which makes them difficult to detect in general.

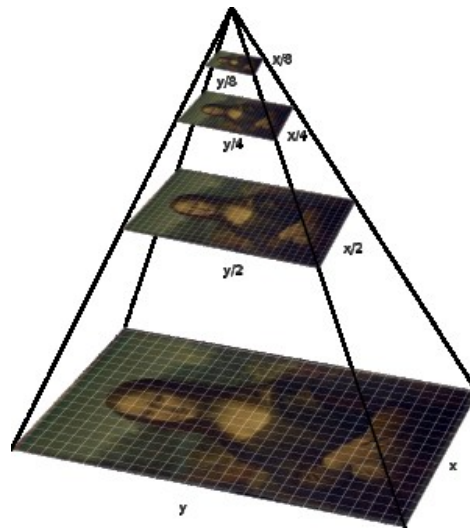


Figure 2.4.5 Image pyramid example with scale set to 50% [54]. The input image at the bottom of the pyramid is the original size and in this instance the image is downsampled by 50% for each step. Each image in the image pyramid is scanned with the detector window to determine if the object of interest is present or not.

## 2.5 EXISTING LITERATURE WITH RESPECT TO POTHOLE DETECTION

### 2.5.1 DETECTION OF POTHOLE VIA VIBRATIONS

The simplest method for detecting potholes can be achieved by detecting the vibrations that a vehicle undoubtedly experience when it encounters a pothole. These vibrations can easily be measured by equipping the vehicle with an accelerometer [56]. This method for detecting potholes is only successful in conjunction with additional equipment that must log the position of the detected pothole via a GPS module. A form of communication (such as a GPRS module for example) is then necessary that can communicate this information to other vehicles/persons of interest before the pothole detections can be useful.

It is also possible to develop a pothole detector that only uses the accelerometer of a smartphone to detect potholes [57]. The on-board GPS and GPRS modules of the smartphone can then be used to upload the pothole detection to a central server which can distribute the

## CHAPTER 2 – LITERATURE REVIEW

---

information accordingly. Smart phones have also become widely available and dropped substantially in price [58] making it a viable alternative option. However, from literature, it seemed that it has not yet been investigated how important the placement of the smart phone is inside the vehicle. It is possible that the built-in accelerometer would give a different reading if it was fixed to the vehicle, laying unfixed somewhere in the vehicle or when it is simply inside a passenger's pants pocket or handbag. Further study is therefore needed in this area as this could create a major disadvantage if only smart phones are used.

In general, it is inexpensive to use vibrations to detect potholes and the solution would be even less expensive if it is assumed that most users already own a smart phone with accelerometer/GPS capabilities and it is only necessary to develop a cell phone software application. In [57], the average detector accuracy for events such as large potholes, small potholes, pothole clusters, gaps and drain pits is reported as 92% at best. Note, however, that by using this method, it is impossible to distinguish between the vibrations created by hitting a pothole or a drain pit.

Using only the measurements by an accelerometer it is clear that this method has an obvious downfall namely, for every pothole detected at least one vehicle must hit it. This is not an ideal situation as it is a very real possibility that the initial vehicle that detected the pothole sustained unwanted damage. Instinctively, a driver could also be inclined to swerve and try to avoid the pothole, which could minimize the new pothole detections in the system. It can also become a very complicated situation to distinguish between a pothole and the slamming of a door or encountering speed bumps [57].

### 2.5.2 VISUAL APPROACH

From analysing the literature it was found that using some type of visual approach that incorporates the use of at least one camera was the most popular choice for pothole detection. A visual approach is suitable, because generally, a pothole will look different in appearance when compared to the rest of the road/background to an observer. Visual cues are the only sensory input needed for a human to detect potholes with an extremely high accuracy rate when the human is giving their full attention to this task provided that the lighting conditions are reasonable. The complex task of visual object detection is relatively easy for humans compared to how difficult it is to teach a computer to “see”. For a child it is very easy, for example, to identify and classify a

---

## CHAPTER 2 – LITERATURE REVIEW

---

variety of airplane models as airplanes after only being exposed to a few examples of airplanes. To achieve the same result by using a computer is at the very least, a daunting task.

An important aspect of pothole detection is when one operates outside the hypothesis that only one pothole will be present in an image at any given time. The reality is that it is quite likely that there could be more than one pothole present in an image and it must be decided if and how this problem would be solved.

### 2.5.2.1 BASIC DETECTION TECHNIQUES

A very simple method for the detection of *simulated* potholes was presented in [59]. The problem of pothole detection was oversimplified in [59] as the specifications dictated that all of the encountered potholes were 2 feet ( $\approx 61$  cm) in diameter and was a distinctive white colour as opposed to the standard dark appearance of a pothole. These simplifications led to the advantage of using a monochromatic camera, which reduces the data per image that is generated by the camera substantially.

The approach begins by plotting the brightness histogram of an image. Using this information, a suitable threshold value can be chosen that will result in a binary image. Due to the vast colour difference between the road and the simulated white pothole, the pothole region can be identified within the thresholded image by detecting the contour of the suspected pothole region. Contours are then detected in the binary image and only those contours that has a diameter with a ratio of circumference close to  $\pi$  are considered.

Images in [59] were processed with an ISCAN RK-446-R image-tracking device which detects the centroid of the brightest or darkest region in a window. The device then outputs the x- and y-coordinates of the centroid as well as the size information of the region. It is clear that using an imaging board would present certain advantages such as removing some of the computational burden from the primary data processing device. The algorithms used by the board are also optimized for efficiency. The imaging board also allowed for the quick implementation of standard libraries which reduced the development cycle significantly.

Due to the simplifications made, however, the approach described is not applicable for a real-world solution but a few good key concepts could be extrapolated from this process and incorporated in a more suitable manner.

---

## CHAPTER 2 – LITERATURE REVIEW

---

### 2.5.2.2 POTHOLE DETECTION VIA TEXTURE, SHAPE AND DIMENSION

In [60], potholes were detected by segmenting an image using its distress criteria. A DFS (Distress Frames Selection) algorithm was developed in [61] and applied to images containing roads. The images that the algorithm finds to contain distresses are then fed to a second algorithm called CDDMC (Critical Distress Detection, Measurement and Classification). This algorithm detects potholes based on their texture, shape and dimension. It is possible to detect potholes by investigating the texture within a pothole as this differs from a normal “healthy” section of road. This is because there are often loose broken up parts of road inside the pothole itself. The paper assumes that a pothole is usually circular in appearance and both this and the assumption that the pothole’s texture differs implies that the detector will detect potholes that are very close to the camera that is used. At a distance further away, a pothole appears to be a horizontal line. The algorithm uses the dimension of the detected disturbance to distinguish between cracks (which seem like lines) and potholes. Furthermore, the images are categorised as images containing potholes, cracks, patches and without distress respectively.

The CDDMC algorithm starts by extracting only the blue channel of an image and processing this channel. A median filtering is then applied to the blue channel which acts as a blurring method for removal of noise in the image. Adaptive thresholding based on the weighted mean produces a binary image which clearly highlights areas that differ from a normal section of road. Additional noise removal is achieved by opening the image and then closing the image, which are a combination of erosion and dilation operations. Connected component labelling a.k.a. contour detection is then applied to the image. The size, circularity and diagonal width of each contour is then calculated. Based on certain size, circular and width criteria, the distresses are labelled. The method will then store the image in the appropriate category folder.

The implementation of the system was based on OpenCV 1.0 and Microsoft Visual Studio 2008 (C++). The images to test the algorithm were obtained with two different cameras in two different setups. In the first setup, 283 monochromatic images were captured that contained a stretch of road of 3 m in length and 2.5 m in width. This camera’s resolution was set to 1280 x 960 pixels. The second set of images obtained (992 images, at 640 x 480 pixels) were in colour and contained stretches of road that were 1.5 m in length and 2 m in width. These images only contain the road surface which reduces the chance for false detections to occur. The results obtained are presented in Table 2.5.1. From the results it is clear that both potholes and cracks can be detected

## CHAPTER 2 – LITERATURE REVIEW

---

with good results, however, patches are more difficult to detect. This could be because (except in texture), the patch emulates the same shape as a pothole and it is difficult to distinguish between the two. If the algorithm was upgraded to investigate the texture within a detected circular contour, it would be possible to distinguish between a pothole and a patch.

Table 2.5.1 CDDMC algorithm results. Results found in [60].

Distress type in image	Accuracy	Precision	Recall
Potholes	97%	95%	81%
Cracks	94%	94%	98%
Patches	90%	8.5%	19%

### 2.5.2.3 POTHOLE TRACKING

Depending on what the needs are of a pothole detection system, it could be decided that it is important to exactly count the number of potholes present on the road. If this is the case, it is sensible to first detect a pothole and then track the detected pothole in subsequent images. Computational processing could possibly be saved in this process as well, if the tracking algorithm is less computationally costly than the detection algorithm itself.

A method utilizing this approach was presented in [62] and its experimental setup can be found in Table 2.5.2. In [62], high speed cameras are used to capture videos that contain roads including a number of potholes. The vehicles themselves form a network that detects potholes and uploads the information onto a central server where the images of the potholes can be assessed by maintenance workers. The experimental setup consisted of a camera that was mounted on a robot which then drove around and provided the necessary footage.



**CHAPTER 2 – LITERATURE REVIEW****Table 2.5.2 Experimental Setup Done by Koch and Brilakis [62]**

<b>Specification</b>	<b>Value</b>
Processing Software	MATLAB v7.11.0
Robot CPU	Viliv S5 Ultra Mobile PC
Camera	HP Elite Autofocus Webcam
Camera resolution	640x480
Camera image rate	30 fps
Camera mount angle	45 degrees facing downwards
Camera mount height	60 cm

To classify potholes in an image the following criteria was used:

- Potholes generally have shadows that are darker than the areas around it
- Due to the view from a vehicle, potholes appear to be elliptical in shape
- The texture of a pothole is much more coarse than that of the tar road

Using histogram thresholding, regions of interest (potholes) can be determined. To check if the second pothole criteria is matched, morphological thinning and elliptical regression is applied. The texture inside the region of interest is analysed and compared to that of an area that does not contain potholes.

In [62] it was identified that to check each image from the footage for potholes was computationally inefficient as the same pothole would be detected several times. Therefore, it was concluded that potholes should be tracked in subsequent images as soon as one was detected.

The texture extraction and comparison method is in essence a method for continually updating the average texture of the road. To determine what the average non-defect pavement is, the centre of subsequent images is cropped (pending the outcome of an image segmentation method checking for defects). To each region four filters are applied which yield feature vectors which each have 5 components. The first component contains the standard deviation of the grey intensities in the region while the other four components indicate the standard deviation of the respective filter's response intensities for non-defect pavement [63]. The vectors are updated for each image to most accurately produce an average texture for non-defect pavement. This is the vector which will determine (texture-wise) whether or not a pothole is detected. This procedure is computationally faster because only two dimensional convolution is calculated over a smaller

## CHAPTER 2 – LITERATURE REVIEW

---

picture region as opposed to the entire image. This method is also better because it is less likely to be disturbed by abrupt surface changes that do not necessarily represent a pothole (darker patch of tar).

It does seem, however, that the implemented algorithm can only detect and track multiple potholes in the event that they all enter an image simultaneously. The reason for this being that the detection algorithm is suspended while the tracking algorithm is in operation. The suspension of the detection algorithm prevents the same pothole from being counted unnecessarily several times over. This was the original goal of the algorithm i.e. to accurately count the number of potholes present in video footage. It is suspected, however, that if this was implemented as a driver warning system, the algorithm would not warn the driver of all potholes in an area where several potholes are present subsequently. The detection of potholes was achieved, however, with a recognition precision of on average 75%.

### 2.5.2.4 POTHOLE DETECTION BY USING STEREO VISION

Pothole detection has also been attempted by using stereo vision as in [64] and [65]. Stereo vision implies that two cameras were used in the experimental setup. When two cameras are used, it is possible through projection theory and triangulation (see Section 3.7 for more detail) to determine the depth within an image. In both [64] and [65], a special rig was built and fit to a vehicle. The two cameras were fixed to the rig in such a manner, that the cameras were perpendicular to the road surface i.e. facing exactly downwards. Both works presented a preliminary study which did not include any tangible results. It is suspected, however, that this method is not suitable for a real-time pothole detection system where the objective is to avoid the vehicle hitting the pothole as the cameras only scan a 2.4 m<sup>2</sup> section of the road [65] at a time.

### 2.5.3 INFRARED APPLICATIONS IN POTHOLE DETECTION

Commercially, the most widely known and affordable infrared sensor system available is manufactured by Microsoft and is called the Kinect. The Kinect is essentially a depth sensor and its properties are listed in Table 2.5.3 [66]. The 830 nm laser diode falls in the near infrared range.

## CHAPTER 2 – LITERATURE REVIEW

Several different researching teams have attempted to investigate the suitability of the Kinect with regards to the detection of potholes [67] as well as in the integration with navigational systems in Unmanned Ground Vehicles (UGV) [68].

**Table 2.5.3 Kinect Sensory Key Properties.** The properties of the Microsoft Kinect was found in [66].

Specification	Value
Laser diode wavelength	830 nm
Laser diode output power	60 mW
Image resolution	640 x 480
Frame Rate	30 Hz

The work done in [67] pertained to the detection of potholes by integrating a camera and a Kinect. The GPS coordinates and the severity of each pothole is also recorded which will allow technical/maintenance staff to properly plan maintenance schedules. The system was required to account for a maximum vehicle speed of 60 km/h and to record a photo, the location and 3D point cloud data of an encountered pothole. The location was then uploaded via the Google Maps Application Programming Interface (API).

This paper also showed results that provided depth information regarding potholes during the daytime for specific selected still images. At 60 km/h, less than two point clouds per second will be produced by the Kinect. The high speed camera used is the DFK22AUC03 with an adjustable image rate (60 to 150 images per second). The maximum image resolution is 744 x 480 but is limited to 320 x 240 at the higher image rates.

The camera and Kinect was connected to the rear end of a vehicle at the tow bar. The software that was run on the computer is the open-source Robot Operating System. The main method to detect potholes was edge detection using the high-speed camera. If a pothole is detected via the high-speed camera, the point cloud data and GPS coordinates of it is stored. Only then, is the point cloud data analysed with the random sample consensus which will determine the plane of the road surface. Anything depressions/protrusions that are not in the plane is concluded to be a pothole.

The paper concluded that the edge detection on the high speed camera proved to be rather accurate at vehicle speeds of 60km/h (percentage was not given).

In [68] it was found that the Kinect was unable to provide any depth information in direct sunlight. An investigation as to why this is the case revealed that it because the radiation from the sun is quite high in the infrared region outdoors. The “noise” from the infrared provided by the

## CHAPTER 2 – LITERATURE REVIEW

---

sun is too high which completely overshadows the output of the Kinect's laser emitter rendering it useless in direct sunlight.

The author of [67] was contacted via email to determine how it was possible to use the Kinect in broad daylight. The author responded and stated that the Kinect seemed to only work outside during overcast conditions but not in direct sunlight.

### 2.6 CONCLUSION

This chapter included all of the theory relevant to this project. The image processing, features, machine learning and applicable pothole papers have been discussed.

In Section 2.2, the HSV colour space was deemed to be a better option in image processing tasks as opposed to the RGB colour space as it separates the luma and chroma information better. It was also highlighted that the Canny edge detector operates in conjunction with a Gaussian kernel. In Section 2.3, the LBP and HOG features were discussed. It was shown that LBP features are invariant to lighting conditions which could make them suitable for pothole detection. HOG features are also light invariant and are more robust in terms of an object with a wide variety of shape than LBP features. Section 2.4 discussed the machine learning algorithms used in this project as well as their application considerations. The specific algorithms that were discussed are the cascade and SVM classifiers. In Section 2.5, the papers relevant to pothole detection indicated that the most common method for detecting potholes is by using a single camera.

# CHAPTER 3

## DATA COLLECTION AND PREPARATION

### 3.1 INTRODUCTION

This chapter describes how the data for the project were collected and processed. Section 3.2 discusses where and how the pothole data was collected. In Section 3.3 the two different types of scenarios where potholes occur in this thesis is discussed. The pre-processing steps that were performed on all of the input data is given in Section 3.4. For both training and testing purposes the data needed to be labelled and is discussed in Section 3.5. The exact number of samples used in this study for each classifier under different circumstances are given in Section 3.6. From visually inspecting the data, it was observed that the appearance of a pothole differs dependant on the distance that it is from the vehicle. Therefore, to identify and analyse the problem with respect to this, it was necessary to use an algorithm that can identify the depth within a monocular image. This work is discussed in Section 3.7. Lastly, a conclusion for the chapter is given in Section 3.8.

### 3.2 DATA COLLECTION

To date there is no dataset available online for pothole data specifically. Therefore it was necessary to identify an area in South Africa where there are many potholes so that the maximum amount of data could be collected quickly and reliably. The Vaal Triangle area in Gauteng was identified as containing a large number of potholes [69] and therefore the data was collected from that region. In addition, a small amount of the data (less than 5%) were also collected around the Stellenbosch and Somerset West region, however, there are very few potholes present in this region. The entire data collection set contains 47 804 images. As the vehicle was driving, the potholes were also manually counted yielding initial estimates of roughly 800 potholes, 100 pothole clusters and 2 craters in the data. A pothole cluster is defined as a location where 5 or more

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

potholes are closely spaced enough that all of the potholes are present in a single image. A crater refers to a section of road where the road has deteriorated so badly that an entire lane is missing and it is necessary to drive onto the oncoming lane's section of road to continue down the road. The craters were not considered to be typical potholes and were excluded from the training/test datasets.

To collect images of potholes, a GoPro Hero 3+ was used. This is because the camera is designed to produce sharp images/videos while moving [70, 71]. Therefore, it is not necessary to deblur the images in the initial pre-processing steps, which potentially saves processing time. The camera was setup in its time lapse mode at an interval of 0.5 seconds/image and produced images at a resolution of 3680 x 2760 pixels.

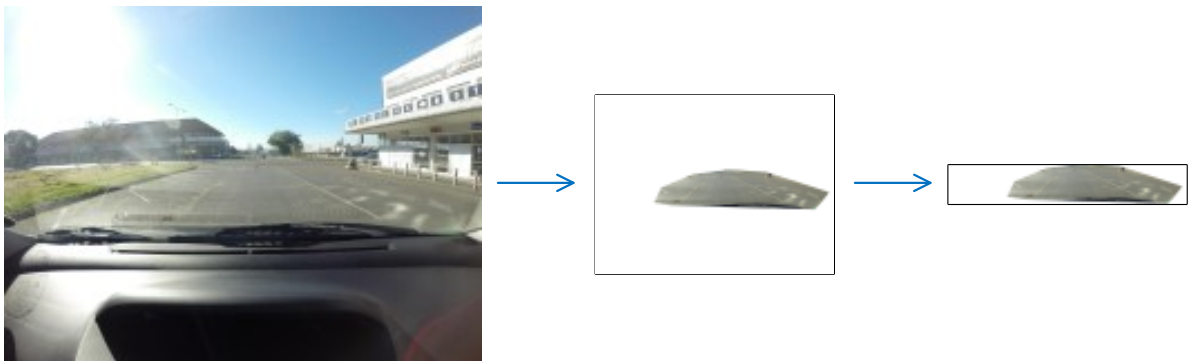
The data acquisition was accomplished by attaching the GoPro camera to the inside of the front windscreen of a vehicle, facing the road. The camera was mounted in such a manner that the optical axis of the camera was as close as possible to being parallel to the road surface. The vehicle was then driven at an average speed of 40 km/h, although there were periods where the vehicle would be accelerating/decelerating when approaching intersections.

### 3.3 DATA SET SELECTION

The task of pothole detection is not trivial. Therefore, to investigate the possibility of pothole detection, the task was considered in two different scenarios. In the first scenario, it is assumed that only open and well lit roads are being used and is referred to as the simple scenario. Due to the definition of this scenario, a few image processing steps is included in the algorithm (see Section 4.2) to extract the road in an image and to only detect potholes on the extracted region. In the second scenario, a typical real-world scenario is portrayed and is referred to henceforth as the complex scenario. This scenario includes road surfaces where there were shadows falling on the road (usually from tree canopies). Both of these scenarios included images from different times of the day. Also, all of the images varied in terms of the position of the sun relative to the vehicle which could be in front, behind, to the left or right of the vehicle. Both of the simple and complex scenarios have separate training and test sets. A summary of the exact number of samples in each set per scenario is given in Section 3.6.

### 3.4 DATA PRE-PROCESSING

The steps taken to pre-process the data for each scenario described in the previous section is indicated in Figure 3.4.1. The simple scenario and complex scenario pre-processing steps are indicated in Figure 3.4.1(a) and in Figure 3.4.1(b), respectively. All of the data used in the study were downsampled by a factor of two, thereby implying that the resolution of the images become 1840 x 1380 pixels. This reduces the computational time of the algorithms.



(a)



(b)

Figure 3.4.1 Input data pre-processing. The pre-processing required for the simple scenario and complex scenario are indicated in (a) and (b), respectively. In both scenarios the initial input figures are downsampled by a factor of two and then cropped to a specific region where the road is estimated to be. In the simple scenario, the road is extracted first before cropping.

In an object detection application such as the one for this thesis, it is necessary to maintain a resolution that is as high as possible because at further distances from the vehicle, potholes can

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

be as small as 7 pixels in height and 14 pixels in width. Therefore, a higher resolution ensures that potholes can be distinguished as an object in the road at larger distances from the vehicle. There is, however, a trade-off between the maximum size of the input images and the computational time it requires to execute the algorithms and this aspect must be kept in mind when a final decision is made with respect to the input image resolution.

After the data has been downsampled, it is cropped to a region where the road is expected to be and can be seen in the figures on the right hand side of Figure 3.4.1(a) and Figure 3.4.1(b). This region was chosen to be 300 pixels in height. This measure, can be interpreted as a form of segmentation and is often used to simplify object detection problems [72, 73, 74]. This segmentation step also reduces the computational requirements of the detection problem substantially. In the case of the simple scenario, an intermediate step is inserted that extracts the road surface in the frame. This step is not included in the complex scenario for the LBP cascade classifier and SVM with HOG features classifiers as the road extraction algorithm often fails under mixed lighting conditions. Therefore, these classifiers' input images contained additional foliage/buildings along the sides of the images. However, as it forms part of the image processing pothole detection algorithm, it is included for the complex scenario image processing detection algorithm.

Finally, note that because all of the classifiers in this thesis only use the grey channel of the image in the detection procedure, all of the input data (whether positive or negative samples) were converted to greyscale.

### 3.5 DATA SET ANNOTATION

The cropped images containing potholes described in Section 3.4 were annotated by using the tools that were obtained from [75], although the latest version of OpenCV now includes annotation software for this purpose. The annotation process involved manually dragging (with the computer mouse) a rectangle around each pothole in each frame. The rectangle serves as a bounding box for the pothole and contains the x- and y-coordinates of its top left corner, as well as its exact width and height.

It is observed that potholes are not like humans in that humans usually have a relatively fixed aspect ratio in terms of width and height [35]. Potholes vary greatly in both aspect ratio and shape not only individually but also due to their change in physical appearance in the road as the



---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

vehicle/camera approaches the pothole. Note that humans usually stand upright in image classification problems, however, potholes occur in the plane of the road and therefore, not perpendicular to the road surface. Machine learning classifiers that use sliding detector windows (discussed in Section 2.4.4) rely on objects/input data that has a fixed width and height ratio. The processes to normalise the input data to a fixed width and height ratio for the machine learning classifiers are described in Section 3.5.1 (positive samples) and Section 3.5.2 (negative samples).

### 3.5.1 POSITIVE SAMPLE PROCESSING

All of the positive samples (potholes) were processed as discussed in Section 3.4. The procedure to normalise the positive samples follows here. First it was determined whether the bounding box for a pothole fits within the detector window i.e. if the pothole is smaller than the detector window in both width and height. If the bounding box fits, the detector window is centred on the positive sample's centre point and the pixel region of the detector window is cropped as a positive sample to be used in the training process of the machine learning algorithms. If the bounding box of the positive sample does not fit, the smallest integer scaling factor necessary to increase the detector window to accommodate the pothole's bounding box is determined. This larger detector window's pixel region is cropped around the pothole and resized to the original size of the detector window, thereby ensuring that all of the input positive samples have the same size.

The above mentioned procedure will result in cropped pothole samples that will include some regular road regions in the samples. As HOG features were designed to detect shapes, the additional road information will allow for a better shape detection as opposed to a pothole that is cropped to its exact dimensions.

### 3.5.2 NEGATIVE SAMPLE PROCESSING

Due to the nature of the functions to train the classifiers in OpenCV, the negative samples for the different types of classifiers used in this thesis needed used to be processed differently and is explained in Section 3.5.2.1 and Section 3.5.2.2.

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

For the simple scenario, additional cropping of the image was done so the image would only be as wide as the extracted road section (as opposed to 1840 pixels wide). This is done to ensure that a higher percentage of the samples included the road surface.

### 3.5.2.1 CASCADE CLASSIFIER NEGATIVE SAMPLING

For the cascade classifier, the full cropped images (1840 x 300 pixels) that contained no potholes were given as input negative samples for training. The `train_cascade` function automatically converts the input images to greyscale (if the images have not been converted to greyscale already) and subsamples them to acquire the negative data for training [76]. These subsampled negative images are only available in memory and are not written to file.

### 3.5.2.2 SVM CLASSIFIER NEGATIVE SAMPLING

When training an SVM classifier in OpenCV, it is necessary to provide the algorithm with negative samples that have been sampled to the correct detector window size and have been converted to greyscale. Therefore, the images that did not contain potholes were sampled by selecting a random number of samples per image, at random points in the image and at a random integer of the detector window size (up to 3 times). If the sample was randomly selected to be an integer multiple larger than the detector window, the sample was scaled down to the size of the original detector window before being included in the negative dataset. In this manner, the aspect ratio of the random negative sample is kept constant and equal to the detector window aspect ratio.

## 3.6 SAMPLE SUMMARY

The exact number of positive and negative samples that were used in training are presented in Table 3.6.1. As there is no training procedure (such as for the machine learning algorithms), a small subset of the training data that were used for the machine learning classifiers were utilized

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

as cross-validation data to perform manual parameter tuning for the image processing method. When training classifiers, as much training data as possible should be used to ensure that the classifier has a large enough database from which to build a model that can generalise most accurately.

Table 3.6.1 Sample summary

	Machine learning algorithm	Positive #samples	Negative #images	Negative #samples
<i><b>Simplified training data</b></i>	<b>Cascade classifier</b>	2350	3399	N/A
	<b>SVM</b>	2350	3399	13291
<i><b>Complex training data</b></i>	<b>Cascade classifier</b>	3074	6192	N/A
	<b>SVM</b>	3074	6192	12519

However, some data also needs to be kept separate so it can be used for testing. Therefore, the number of potholes in the simplified test set is 1219 (from 650 images) and the number of potholes in the complex test set is 1322 (from 604 images). As the entire cropped image is presented to all of the algorithms and not only selected areas within the image, any other area in the cropped image is considered to be a negative sample. Also, the machine learning classifiers utilise image pyramids (see Section 2.4.4) in their detection algorithms which drastically increases the number of windows scanned for potholes. Each window that does not contain a pothole is a negative sample. Thus, it is possible that an image containing only one pothole, could have 5 positive samples and thousands of negative samples due to the effects of the sliding window and image pyramiding functions.

## **3.7 DEPTH ESTIMATION**

### **3.7.1 INTRODUCTION**

Depth estimation is not a new problem and many approaches have been developed for this task. These techniques employ various assumptions, so the appropriate depth estimation technique depends on the application domain. Major factors determining the applicability of an approach for a domain are the financial and computational cost of implementing it. This section of the thesis considers the problem of rapidly determining the distance from a vehicle to a pothole by depth estimation in high-resolution images. Ideally, the solution should be suitable for deployment with the camera for real-time depth estimation of potholes and other obstacles.

Detecting distant potholes is considerably more challenging than detecting nearby potholes. The initial motivation for developing the depth estimation techniques described here was attempting to gain a better understanding of this aspect: by determining the distance of each pothole in set of images, it is possible to refine the results based on these distances. This allows one to quantify the performance degradation of the classifier as the depth increases, and to verify whether the degradation is monotone. In addition, if the results indicated that depth is indeed a factor in the performance of the classifier, the depth information can be used to construct classifiers for potholes at different ranges - it may be possible to combine these to improve overall pothole detection performance.

For the work presented in this section, the following factors were relevant. Potholes occur in countries and states that are subject to heavy rainfall, poor drainage, harsh winter weather and frequent freezing and thawing of road surfaces. However, in less affluent countries, potholes are usually not repaired in a timely fashion and therefore it was important to identify a cost-effective approach for depth estimation. This limitation prevents the use of multiple cameras or expensive laser mapping technology. Therefore, stereo vision techniques fall outside the scope of this project. Additionally, the lack of laser mapping for establishing ground truth depths for objects necessitated the development of a low-cost but accurate approach to validate the depth estimation results. Finally, it was important to find a solution that could potentially be used in a real-time environment allowing a full real-time pothole detection and depth estimation solution. Approaches that enable real-time depth estimation within a road are also applicable to other vehicle automation tasks, such as incorporating rear-end collision avoidance into cruise control applications.

### **3.7.2 EXISTING WORK**

Depth estimation as applied to the work in this thesis is novel, however, depth estimation itself is widely applicable in practice. In robotics, it is crucial for a robot to determine its distance from obstacles in its path [77]. Another application pertains to forensics where only a single stationary camera is available and it is necessary to determine at what distance a person of interest was at a particular time or the height of a particular person in the image is required [78]. Motion capture has also become part of many computer games and films, and by using triangulation it is possible to determine the distance of actors and their limbs relative to other actors [79].

Depth estimation approaches are generally either active or passive. Active techniques generally analyse reflections of sound or light waves emitted into the scene to estimate depth [80]. An example of this is given in [77] where a laser light and camera are combined in such a manner that the camera can detect the laser light reflecting from obstacles. The camera was used to capture images at regular intervals and determine the centre point of the laser light pointing at an obstacle. The perceived shift of the laser light relative to the movement of the robot was then used to estimate depth with good accuracy. However, active approaches rely on the availability of additional equipment.

Passive techniques perform depth estimation by analysing one or more captured images of a scene. Techniques have been developed for estimating depth from video feeds, estimating positional information from multiple images of the same scene (most commonly for stereo vision applications), and monocular depth estimation, where only a single image is used for depth estimation. The classical approach in all these cases involve trigonometric and (typically projective) geometric computations, incorporating whatever additional information is available from the particular domain. For the task of pothole detection, only one camera is used, and using video feeds would only be feasible if an object tracking system were available for the potholes. Thus, the work presented here focus on monocular vision approaches.

Two papers are of particular interest with respect to depth estimation of objects in a road using monocular techniques. In [81], a single camera was used to estimate the depth of a vehicle on an (assumed) planar road surface. The approach relies on the assumption that the camera's optical axis is parallel to the road's surface and edges. A geometric approach is applied that combines the pinhole model and ad-hoc error corrections presumably obtained from empirical measurements. An accuracy of 96% was reported over a distance of 70 m, but the methodology

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

for obtaining the corrections is not presented, so that it is unclear what must be done to obtain this accuracy.

In [82], monocular depth estimation of vehicles on the road was also tackled using a pinhole camera model. In particular, the depth of the point where the vehicle meets the road was estimated. As in [81, 83], a planar road surface and a parallel installation of the camera's optical axis to the road surface and edges are assumed. The work also included a method for determining the range rate that was based on scale change.

As an alternative to the classical geometry-based depth estimation techniques discussed above, it is worth noting that work has also been done on using probabilistic modelling of scenes and the structure of images for extracting depth estimation from single images. The most notable approaches in this direction make use of Markov random fields; e.g. [83, 84, 85]. These techniques are very versatile, generating range images for scenes appropriate to the situations they model without any major assumptions about the camera setup. These techniques are less appropriate for use in this project for two reasons. First, a suitable probabilistic model of a road and its contents is needed to apply the model; second, even with a suitable model, the accuracy of these is generally not as accurate as the geometric techniques discussed previously, so the latter should generally be applied when the relevant information for applying the geometric techniques is not available.

### 3.7.3 RELEVANT LITERATURE

This section discusses aspects of the pinhole camera model and camera calibration process relevant to the rest of the work presented in Section 3.4.

#### 3.7.3.1 PINHOLE CAMERA MODEL

For the approaches used in Section 3.4, the camera is modelled using the pinhole camera model [86] as illustrated in Figure 3.7.1.a, and only rays of light coming through the pinhole, are considered. The ray of light leaving the top of the tree in the figure projects onto a pixel on the top of the image plane  $I_2$  and a ray of light from the bottom of the tree projects onto a pixel at the bottom of the image plane  $I_2$ . Similarly, other rays of light between these two rays project onto the

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

image plane, and if the camera is aligned to the tree such that the middle of the lens is aligned with and pointing directly toward the middle of the tree, the ray of light from the middle of the tree will travel through the pinhole parallel to the ground. The distance between the image plane and the pinhole is known as the focal length  $f$ .

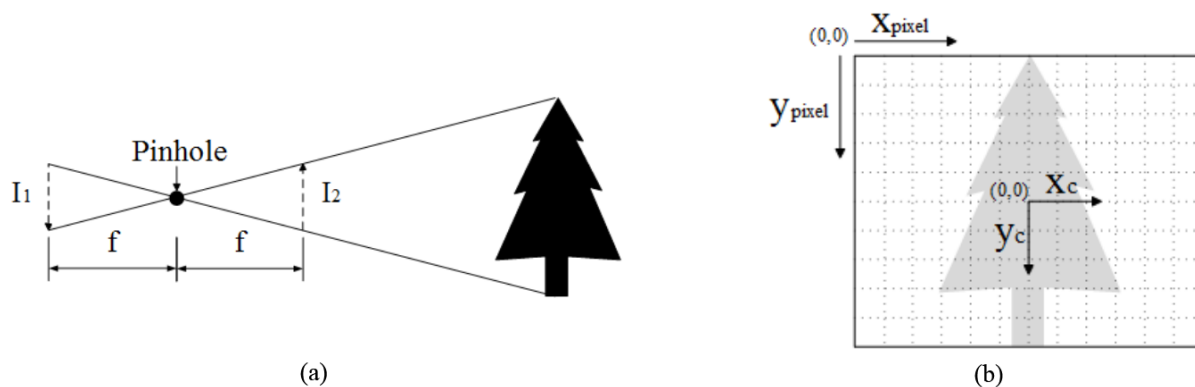


Figure 3.7.1 Illustration of (a) the pinhole camera model and (b) the corresponding camera plane and image co-ordinate systems.  $I_2$  represents the image plane, and the focal length  $f$  is the distance between the pinhole and the image length.

The plane  $I_1$  falls precisely on the internal CMOS sensor of the camera, and detects an inverted (upside-down) smaller image of a tree in this case.  $I_2$  is symmetric to  $I_1$  about the pinhole, so that the camera captures the image on the image plane by inverting (in software) the image captured on the CMOS sensor at  $I_1$ . A different view of the image plane is given in Figure 3.7.1.b, which clearly indicates the relationship between the pixel coordinates ( $x_{\text{pixel}}$  and  $y_{\text{pixel}}$ ) and the x- and y-axes of a camera coordinates ( $x_c$  and  $y_c$ ). The z-axis of this system is then orthogonal to the image, and the z-coordinate of an object in this coordinate system thus represents its depth in the scene. Thus, estimating depth using this method involves directly estimating the z-coordinate in the camera world corresponding to a pixel location.

Since the origin in the camera coordinate system is at the pinhole, if a depth of 5 meters is calculated, it will refer to a distance of 5 meters in front of the camera. This is suited for the application, where the camera is mounted inside a vehicle and the camera reference point needs to be dynamic in relation to the movement of the vehicle on the road.

In general, the mapping from coordinates used for specifying a location in the real world (called world coordinates) to camera coordinates is a projective transformation that can be described by a projection matrix  $\mathbf{P}$ .  $\mathbf{P}$  depends on the camera's intrinsic and extrinsic properties as shown by equation (3.7-1). The assumptions made about the camera installation (see Section 3.4.4) simplify the situation by fixing the extrinsic parameters: the assumptions set the rotation matrix  $\mathbf{R}$

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

to the 3 x 3 identity matrix and the translation vector  $\mathbf{t}$  to the zero vector, i.e. it is assumed that the camera and world coordinate systems are identical. The intrinsic parameters of the camera are represented by the calibration matrix  $\mathbf{K}$  - these depend on the camera and its configuration. The next section discusses camera calibration, a process for estimating the calibration matrix.

$$\mathbf{P} = \mathbf{K}[\mathbf{R} | \mathbf{t}] \quad (3.7-1)$$

### 3.7.3.2 CAMERA CALIBRATION

Various forms of the calibration matrix can be found in [86]. One of the most common representations is given in equation (3.7-2), where  $f_x$  and  $f_y$  represent the focal point in the x and y-axis respectively, and the centre point of the CMOS sensor of the camera (also referred to as the principal point) is represented by  $u_x$  and  $u_y$ .

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.7-2)$$

Setting  $\mathbf{K}_{0,1} = 0$  indicates that the skew of the sensor is presumed to be zero i.e. the image is not distorted in a diagonal manner. In [84] the camera is not calibrated and is chosen as  $(u_x, u_y) = (W/2, H/2)$  where  $W$  represents the width of the image in pixels and  $H$  represents its height. This is usually an acceptable assumption but the origin can vary slightly depending on the manufacturing quality of the camera. For the purposes of this project, the camera was calibrated, which yields the most accurate calibration matrix for the camera when needed.

The camera was calibrated with the Camera Calibration Toolbox for Matlab [87], using a board with 100 mm x 100 mm squares to enable more accurate calibration at the larger distances relevant to this task. The (estimated) board positions used for calibration are displayed in Figure 3.7.2. Note that the plane described by  $y = 0$  in this space is not the road surface itself, because the camera (which is at the origin) is mounted within a vehicle.



## CHAPTER 3 – DATA COLLECTION AND PREPARATION

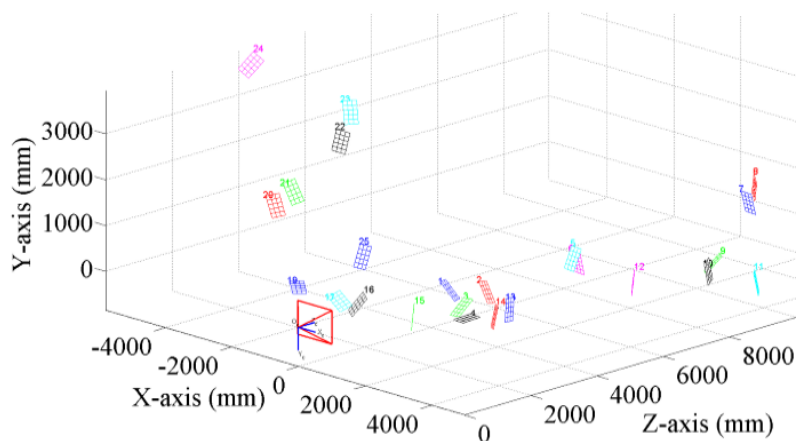


Figure 3.7.2 Camera calibration. Approximate board locations used for camera calibration. The camera is at the origin.

Since cameras in practice make use of lenses that adjust the direction of light travel nonuniformly, the pinhole camera model is not entirely accurate. It is thus generally desirable to remove the fisheye effect (or other perspective distortion) from images in which depth needs to be estimated, in order to maximize the accuracy of the depth estimation results.

In order to obtain accurate distortion parameters, it is necessary to place the board at various points near the edges of the field of view of the camera, where the fisheye effect is most noticeable. The left hand image in Figure 3.7.3 illustrates the fisheye effect of the camera; after calibration the distortion could be mostly eliminated, as seen in the right hand image in Figure 3.7.3.



Figure 3.7.3 Fisheye distortion illustration. Before distortion (left) the straight lines in the figure are distorted and appear bent. After the fisheye distortion correction (right) the distortion is barely visible.

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

It can be seen that the fisheye effect has not been eliminated; however the distortion has been reduced enough for its remaining effects to be negligible in future calculations as was determined from the work done to produce Figure 3.7.4.

To measure the fisheye distortion in the horizontal plane, one of the horizontal lines of the venue where the images were taken was used. The line is indicated by the green crosses in Figure 3.7.3. This line is a straight line, however, due to the fisheye distortion introduced in the image, it can be observed that it is not. The expected straight line was used as a reference and from its centre point, the deviation from this line was measured and a relative percentage error was calculated and is indicated in Figure 5. As can be seen, the corrected fisheye image's graph indicates an error of less than 5% towards the edge of the image.

However, the effects of not performing fisheye correction before depth estimation was investigated and is presented in Section 3.4.5.

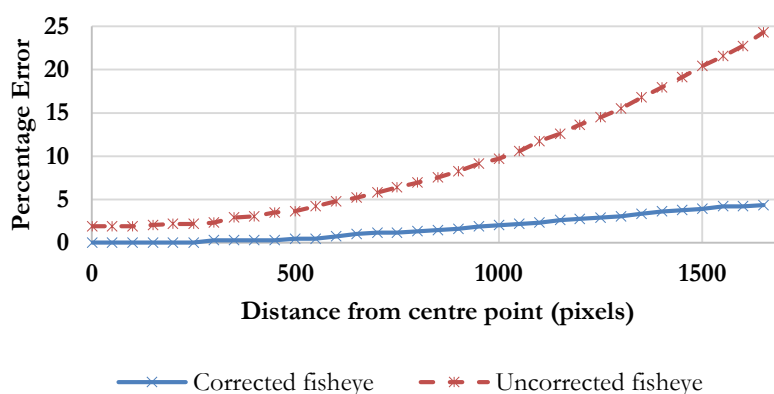


Figure 3.7.4 Fisheye correction measurements. The measurements were taken with respect to the horizontal centre of the image

### 3.7.4 METHODOLOGY

The approaches considered make use of two assumptions that simplify depth estimation. These assumptions are based on the fact that the camera will be mounted in a vehicle and will be facing the road when relevant images are taken. The first assumption is that the road surface is presumed to be flat relative to the vehicle. This should generally be the case below the vehicle, but the assumption may be incorrect for the road section ahead of the vehicle if the road slope changes. In this case, the depth estimation techniques will provide less accurate results. The second

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

important assumption is that the mounted camera's lens is directly facing the road and that it faces the road in such a manner that a light ray through the centre point of the camera lens is parallel to the road surface and edges, thereby implying that there is zero yaw, pitch and roll when the camera faces the road. This is the assumption simplifying the form of the projection matrix by fixing the extrinsic parameters to convenient values, described in Section 3.4.3.2. Note that there are methods for determining the ground plane, such as the work done in [88], if these assumptions are invalid. These techniques fall beyond the scope of the work presented here, which focuses on contrasting the accuracy of depth estimation techniques.

Section 3.4.4.4 describes a simple approach based on the geometry of similar triangles. An approach employing the cross ratio of collinear points is then described in Section 3.4.4.2. Section 3.4.4.3 finally describes a versatile depth estimation which makes use of camera calibration, and only requires one other reference point in the image. This approach essentially determines the location of an object in the ground plane by solving equations involving the projection matrix  $\mathbf{P}$ . This approach is referred to as the pinhole (camera) model approach. The low-cost experimental setup used to compare the accuracy of these methods is given in Section 3.4.4.4.

### 3.7.4.1 SIMILAR TRIANGLES

A common geometric approach to estimating depth of an object directly ahead of the camera relies on the geometry of similar triangles [89] is deduced from Figure 3.7.5. The equation one arrives at is given in equation (3.7-3) where  $z$  is the estimated depth in the image if the focal length ( $f$ ), height of the camera ( $H$ ) and  $y$  pixel value ( $y$ ) at the point on the image plane are known.

$$z = \frac{fH}{y} \quad (3.7-3)$$

This formula can be employed to estimate the depth of an object on the road once the focal length of the camera has been determined: since the height of the camera is known, this formula implicitly maps pixel heights to estimated distances. Two approaches to obtaining the focal length in this work was considered.

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

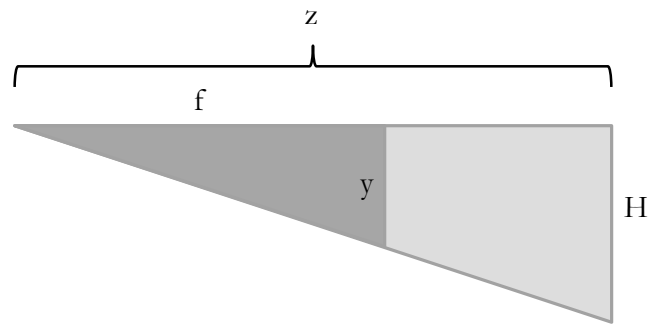


Figure 3.7.5 Similar triangles illustration. The sketch illustrates how the formula in equation (3.7-3) is derived from standard geometry.

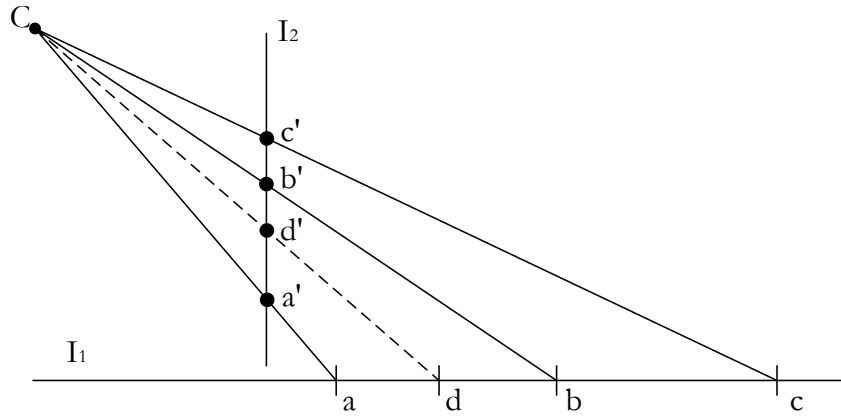
### 3.7.4.2 CROSS RATIO FORMULA

The second approach to depth estimation that is considered makes use of reference pixels corresponding to known distances within the image. The approach, presented in [78], employs the cross ratio projective transformation defined equation (3.7-4) [13]. Here, points a-d refer to homogenous points in a one-dimensional plane  $\mathbb{P}^1$ , so that the right hand side is a quotient of products of matrix determinants.

$$\text{Cross}(a,b,c,d) = \frac{|a \ b| |c \ d|}{|a \ c| |b \ d|} \quad (3.7-4)$$

Consider four rays in a plane passing through a common point C. The projection of these rays on any other plane yields four collinear points, and the cross ratio of these points is invariant to the choice of the plane [90]. The use of this technique is illustrated in Figure 3.7.6, where C is the pinhole from the pinhole camera model. In this figure, the one-dimensional data on line  $l_1$  represent depths in the road directly ahead of the camera and the one-dimensional data on line  $l_2$  are the y-values of the corresponding pixels in the image, with x-coordinate equal to that of the principal point.

## CHAPTER 3 – DATA COLLECTION AND PREPARATION



**Figure 3.7.6 Cross ratio point mapping.** The mapping from  $I_1$  to  $I_2$  projecting the points  $a, b, c$  and  $d$  to  $a', b', c'$  and  $d'$  respectively is an homography through  $C$ , and the cross ratio of sets of collinear points is preserved by homographies.

This technique is applied by determining pixel locations for a number of known distances in the road in advance, and then selecting three equally-spaced distances for use as  $(a-c)$  together with an unknown distance  $d$  corresponding to a particular pixel height  $y$ . It is not required that the reference points be equally spaced, but it simplifies the exposition below. Thus the cross ratio for the data in the one dimensional road plane can be expressed as  $\text{cross}(a, b, c, d) = \text{cross}(0, L, 2L, d)$  and similarly that for the one-dimensional pixel  $y$ -values can be expressed as  $\text{cross}(a', b', c', d') = \text{cross}(y_0, y_1, y_2, y)$ , where  $y_0, y_1$ , and  $y_2$  have been determined in advance. Equating these cross ratios to determine  $d$  yields equation (3.7-5). Note that this approach implicitly measures depths relative to the depth of reference point  $a$ .

$$d = \frac{(y_0 - y_1)(y_2 - y) - (y_0 - y_2)(y_1 - y)}{(y_0 - y_1)(y_2 - y) - \frac{1}{2}(y_0 - y_2)(y_1 - y)} L \quad (3.7-5)$$

For experimental purposes, the pixel values for reference points were determined at 5 m intervals (from 5 m to 30 m), so equation (3.7-5) could be applied with  $L = 5$ . Points  $a-c$  were selected to include the reference points directly above and below the pixel of interest: for depths below 15 m, the reference points are 5 m, 10 m and 15 m. Between 15 m and 20 m, the reference points are 15 m, 20 m and 25 m. Lastly, for depths between 20 m and 30 m, the three reference points are 20 m, 25 m and 30 m.

The approaches described above (using similar triangles or the cross ratio) can only predict distances for points directly ahead of the camera. In order to perform depth estimation for other

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

pixels, only the y-coordinate of the pixel is used. If the camera setup is correct and the perspective distortion has been eliminated, this will give the correct depth - essentially, the depth of a point in the road at a pothole can be predicted.

### 3.7.4.3 PINHOLE CAMERA MODEL FORMULA

The pinhole camera model specifies that the projection of a ray through the pinhole onto the image plane is determined by the projection matrix  $\mathbf{P}$  according to equation (3.7-6), where  $\mathbf{X}_w$  specifies the homogeneous world coordinates of the ray and the pixel coordinates are denoted by  $\mathbf{x}$ .

$$\mathbf{x} = \mathbf{P}\mathbf{X}_w \quad (3.7-6)$$

Using the pseudoinverse  $\mathbf{P}^*$  of  $\mathbf{P}$  (note that  $\mathbf{P}$  is not square), the homogeneous coordinates of the ray passing through a pixel of interest and the pinhole can be determined by

$$\mathbf{X}_w = \mathbf{P}^* \mathbf{x} \quad (3.7-7)$$

To determine the world coordinates corresponding to the pixel in question, it is necessary to determine the point where the ray travelling through this pixel and the plane of the road intersects. After computing  $\mathbf{X}_w$ , it must thus be dehomogenized to yield the desired world coordinates by finding a suitable scalar factor  $\lambda$  in equation (3.7-8).

$$\mathbf{X}_w = \lambda \mathbf{P}^* \mathbf{x} \text{ where } \lambda \in \mathbb{R} > 0 \quad (3.7-8)$$

The derivation to determine  $\lambda$  uses the normal form representation of a plane (equation (3.7-9)) which characterizes the points on a plane in terms of a reference point  $V$  in the plane and the normal  $\mathbf{n}$  of the plane.

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

$$\mathbf{n} \cdot (\mathbf{P}_1 - \mathbf{V}) = 0 \quad (3.7-9)$$

The normal of the road plane is always the same in this application due to the assumptions about the camera setup, and it can be determined by placing the calibration board flat on the road surface, and recording the translation vector and rotation matrix describing the position and orientation of the board. The location the calibration board is placed at serves as the reference point  $\mathbf{V}$  (Note that the thickness of the calibration board must be taken into account).

Next, setting  $\mathbf{P}_1 = \mathbf{X}_w$  in equation (2.4-9) and solving for  $\lambda$  yields:

$$\lambda = \frac{\mathbf{n} \cdot \mathbf{V}}{\mathbf{n} \cdot \mathbf{P}_x^*} \quad (3.7-10)$$

Finally, using this value of  $\lambda$  in equation (2.4-8) yields the estimated world coordinates of the pothole in the road.

$$\mathbf{X}_w = \frac{\mathbf{n} \cdot \mathbf{V}}{\mathbf{n} \cdot \mathbf{P}_x^*} \mathbf{P}_x^* \quad (3.7-11)$$

An advantage of this method is that it explicitly provides the world coordinates corresponding to a pixel and therefore contains information of the relative position of the pothole for all three axes, not just the depth.

### 3.7.4.4 EXPERIMENTAL SETUP

To determine the true depth within the images, a measuring tape was placed on a flat surface in front of the camera such that the tape appears to run vertically in the camera image. Measurements were taken at 1 m intervals between 5 m and 30 m. At each of the one meter intervals between 5 m and 30 m the calibration board was held upright and flush with the surface and photographed. These results are recorded to demonstrate how accurate the pinhole model could be if more scale and orientation information were available at the point of interest. This process was performed using the same setup used for the camera calibration. In order to further

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

quantify the accuracy of the techniques, an additional set of measurements were taken parallel to these, but at a distance of 4.93 m to the left of the camera. The camera was set up again, but not recalibrated, for this process.

In order to minimize the alignment problem between the camera mounted within the vehicle and the centre point of the scene, the vehicle setup was eliminated in the final measurement setup. Consequently, the camera was placed on a tripod at the exact height it would have been in the vehicle.

The images used for this study were captured by the same GoPro Hero 3+ camera that was used to collect the pothole databases.

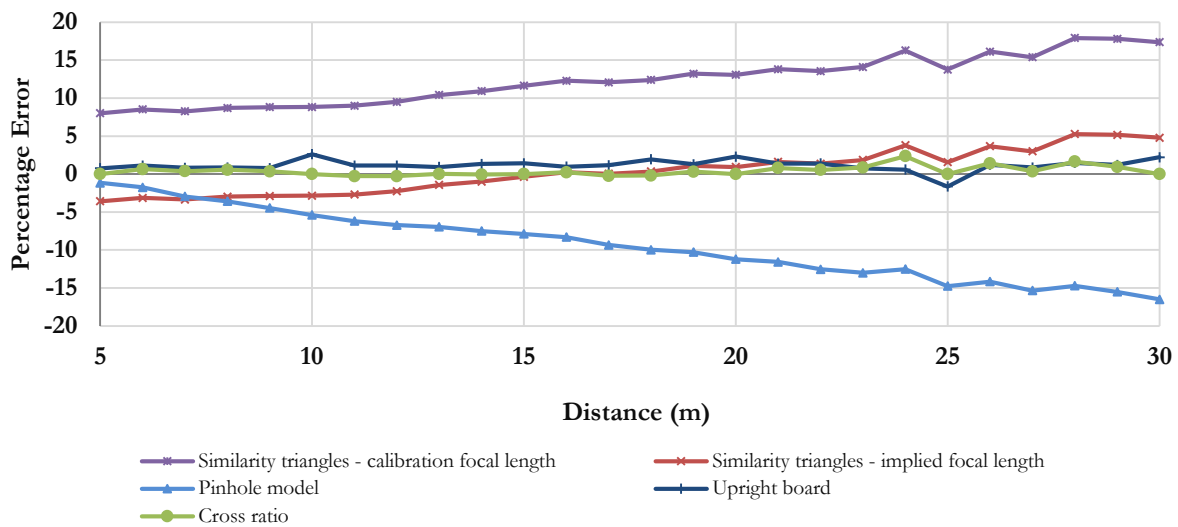
In a real-world scenario, a calibration procedure would be required to mount the camera to the vehicle to ensure the most accurate results. If the pothole detector becomes a commercial product, the manufacturer could develop a calibration device/procedure that would ensure the camera is mounted correctly.

### 3.7.5 RESULTS OF DEPTH ESTIMATION ALGORITHMS

The principal findings of the study are shown in Figure 3.7.7. This figure displays the error percentage in predicted distance at various distances from the camera, when the object is directly ahead of the camera. The values are plotted for each technique. For the similar triangles approach, results are given using the focal length estimated from camera calibration, as well as that obtained by averaging the focal length implied by the reference points used by the cross ratio approach. The pinhole model approach still has good performance for short distance, but the error percentage worsens approximately linearly as the distance increases, resulting in very poor performance for large distances. The similar triangles approach using the focal length from the camera calibration process also fares poorly over the entire range of distance, while the version using the averaged implied focal lengths performed much better. The disparity in these results is surprising, since other aspects (including the “Upright board” measurements discussed below) indicated that the values in the calibration matrix were fairly accurate.



## CHAPTER 3 – DATA COLLECTION AND PREPARATION



**Figure 3.7.7 Results of the various approaches.** The error percentages of the approaches considered at various distances directly ahead of the camera. A negative (positive) percentage error indicates underestimation (overestimation) of the depth.

The cross ratio approach performed best among the approaches (with an error of less than 2% across the entire range), performing even better than the pinhole model approach augmented with additional information from the upright board used during the measurement process. (These results, labelled "Upright board" in the figure, are a dramatic improvement over the baseline performance of the pinhole model.) Omitting the fisheye removal step had a negligible effect on all these curves. These results indicate that for the problem at hand, where distances are estimated between 5-30 m directly ahead of the camera, and have reference pixels of known distance available, the cross ratio formula is the technique of choice. However, objects may not be directly ahead of the camera, which the cross ratio and similar triangles approach expect - the proposal of only considering the y-coordinate may introduce too much inaccuracy. Furthermore, in this situation, if camera calibration is not performed, fisheye removal may not be possible. Finally, one may not have as many reference points in practice as used above. The effect of these limitations are considered next.

Figure 3.7.9 provides results analogous to those in Figure 3.7.7, except that these errors are those obtained for the measurements 4.93 m to the left of the camera. At close distances, this corresponds to a wide angle, so the fisheye effect is much more noticeable for the location of the measurement on the image.

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

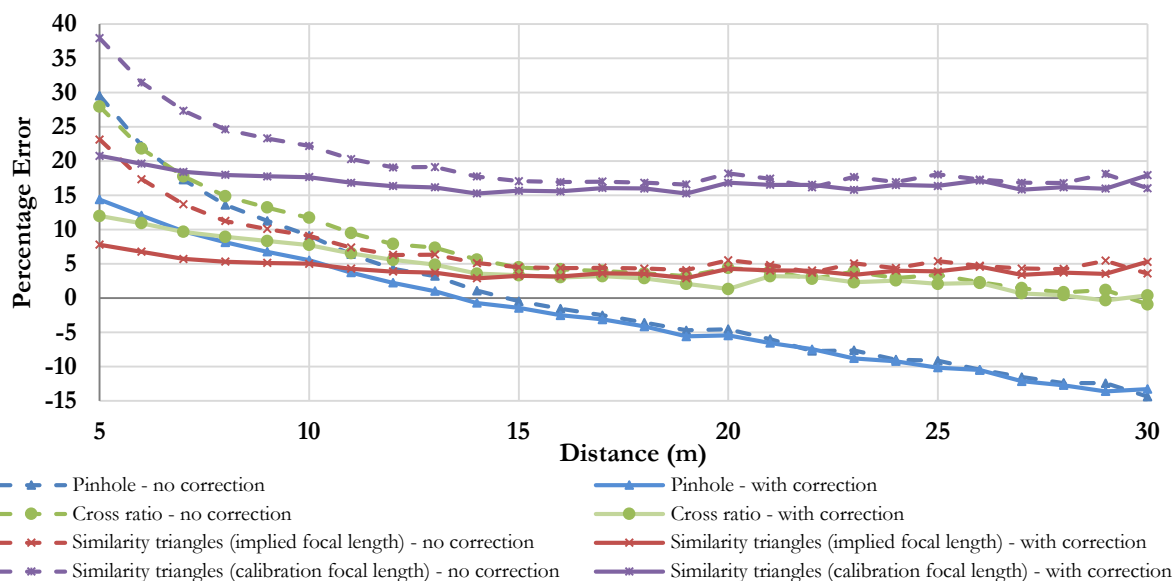


Figure 3.7.8 Results of the various approaches at 4.93 m to the left of the camera. Solid lines indicate error with fisheye correction performed, dotted lines indicate error with fisheye correction omitted. A negative (positive) percentage error indicates underestimation (overestimation) of the depth.

Thus the figure also includes results when fisheye correction is not performed, since the difference is no longer negligible. Finally, at these angles, it is not always possible to reliably estimate the distance using the information from the upright board due to limitations in the camera calibration toolbox, so the corresponding curves for that method are omitted. It is not surprising that the results are considerably worse in this scenario. Consider first the results where fisheye correction has been performed. Once again, the estimated distance for the pinhole model increase too slowly, resulting in a linear decrease on the graph; however, since the model initially overestimates the distance, there is a short window where it performs well. The cross ratio approach begins slightly worse than the pinhole model at 5 m, but becomes consistently more accurate as the distance increases, with less than 5% error for all distances beyond 13 m. Again, the similarity triangles approach using the calibrated focal length performed dismally. However, when this approach used the average implied focal lengths, it performed considerably better than the other approaches for shorter distances, and is only slightly worse than the cross ratio technique beyond 15 m.

Failure to perform fisheye removal in this setting has a notable effect, leading to increases in the estimated distances. The difference in accuracy resulting from the failure to perform fisheye correction reduces as the distance increases (from about 20% at 5 m, to a negligible effect at 20 m). This is because the angle from vertical decreases and the pixels under consideration thus move

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

closer to the centre of the image, where the fisheye effect is weaker. These results indicate that if one's system should be able to estimate distances to objects at wide angles, it may be preferable to use the similar triangles approach (with implied average focal length) rather than the cross ratio approach, since the loss of accuracy directly ahead and at further distances may be countered by improved performance at wide angles.

In order to investigate the effect of the number of reference points used as anchors for the cross-ratio formula, Figure 3.7.9 compares the cross ratio results in Figure 3.7.7 to the performance of the cross ratio approach when only three reference points (the minimum needed for the cross ratio approach) were used. In particular, a 10 m interval was used, with reference points at 5 m, 15 m, and 25 m. Since the error of the techniques at a reference point for this technique is zero, there is more freedom for the technique to introduce error with the wider-spaced reference points, and there is a reduction in performance. However, the performance is still much better than the pinhole model, and comparable to the best other approaches. (Note that it would also be expected that the similar triangles approach using implied focal lengths, to worsen if only given three reference points.)

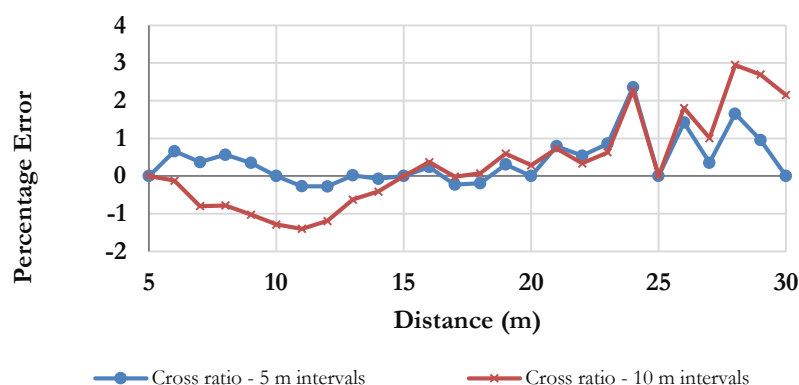


Figure 3.7.9 Cross ratio comparisons. Comparisons between the difference in the error when using 5 m intervals and 10 m intervals

### 3.7.6 DEPTH ESTIMATION AS APPLIED TO POTHOLES?

### 3.7.7 DEPTH ESTIMATION SUMMARY

The work presented here assessed three different approaches to determining the depth estimation problem of determining the distance to an object, such as a pothole on a road. The

---

## CHAPTER 3 – DATA COLLECTION AND PREPARATION

---

pinhole model requires camera calibration to be performed, while the cross ratio approach requires reference pixels corresponding to known distances to be determined. The approach using similar triangles requires the focal length of the camera to be known. This information might be available for some camera models, but might need to be calculated (either as part of a camera calibration process, or from reference pixels as per the cross ratio approach). Furthermore fisheye distortion can impact distance estimation, so it is desirable to remove it if possible; however, this also requires camera calibration.

It was found that for depth estimation directly ahead in the range 5-30 m, the fisheye correction has negligible effect, and the techniques based on reference points were far superior to the pinhole model approach, with the cross ratio approach performing the best directly ahead of the camera. At wide angles (and thus short distances) in this range, the fisheye effect is stronger and the similar triangles approach using implied average focal lengths performs better. If depth estimation in these cases are needed, performing camera calibration to remove the fisheye effect is recommended, and considering a hybrid approach, using the cross ratio approach to get good accuracy directly ahead and at larger distances, and using the similar triangles approach for depth estimates at wider angles.

A limitation of this study is that the results for the pinhole model depend on the calibration matrix, as does the similar triangles approach using its focal length. While care was taken to calibrate the camera accurately, there are always errors in such a process, and it may be that recalibration yields different results. While the results are believed to qualitatively capture the nature of the problem with the pinhole model approach for this task - the distance estimation error percentage increases linearly to unacceptable levels at larger distances - it would be worthwhile to verify this.

Finally, it is noted that if the camera height is changed (for example, by deploying the camera in a different vehicle), the pinhole model and similar triangle approaches can be employed directly once the change in height has been established. On the other hand, the cross ratio approach requires that the location of the reference pixels be re-established. This is an argument in favour of using the similar triangles approach throughout.

### **3.8 CONCLUSION**

This chapter discussed all of the information with respect to the handling of the data. The methodology used to collect the data was given in Section 3.2 which most importantly stated that the footage was obtained by driving around in a vehicle with a fixed camera attached to the front windscreen. Section 3.3 highlighted that there are two different scenarios considered in this investigation into pothole detection, namely a simple scenario and a complex scenario. The data needed to be pre-processed before training or testing could be done and these steps are indicated in Section 3.4. For the same reason, the data needs to be annotated. This procedure was given in Section 3.5. The number of positive and negative samples that were available for each scenario described in Section 3.3 was tabulated in Section 3.6. In order to analyse the classifiers' performance more practically, a novel classifier performance metric was introduced, namely, the distance the object is from the camera. This investigation into depth estimation in monocular images was presented in Section 3.7 which concluded that the cross-ratio formula is the most suitable for the purposes of this thesis. Note, that from studying the available literature on classifiers, it was found that this kind of analysis on object classifiers has never been done before.

---

# CHAPTER 4

## DEVELOPMENT AND METHODOLOGY

### 4.1 INTRODUCTION

This chapter discusses the methodology that was followed to develop the image processing pothole detection algorithms, as well as the necessary design details of the machine learning algorithms.

In Section 3.3, it was discussed that the pothole detection problem is considered for two different scenarios, namely a simplified and a complex scenario. The method for extracting the road surface for the simplified scenario is given in Section 4.2. A step-by-step illustration of the image processing algorithms is given in Section 4.3. To determine the required detector window size a methodology was proposed in Section 4.4. The methods used to find the optimally trained SVM in OpenCV is discussed in Section 4.5.

### 4.2 ROAD EXTRACTION METHOD

The task of identifying occurrences of an object within an image can be simplified by identifying regions of interest in which the object may occur. The process of detecting these regions is known as segmentation [20].

During the initial stages of this thesis, it was found that irrelevant information in an image, such as foliage, can lead to false positives. Therefore, by extracting the road surface using image processing algorithms, and only scanning the extracted area for potholes, the likelihood of detecting false positives can be reduced. This is reasonable since the only region of interest with regards to potholes, is the road surface directly in front of the vehicle. By removing all other objects and foliage outside of the road, these irrelevant areas can no longer interfere with the classification process.

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

Extraction of the road was achieved by using a small rectangular region of interest within the image boundary just above the hood of the vehicle as can be seen in Figure 4.2.1. This region of interest is indicated by the green rectangular area in the image Figure 4.2.1.a. This method assumes that this section will always contain a part of the road if the driver maintains a safe following distance from a vehicle ahead. The area is converted to the HSV colour space and the mean and standard deviation per colour channel of this region of interest is calculated. In order to robustly address the issue of road colour variation within the image, the road colour is modelled separately per channel as lying within three standard deviations of the mean of each channel. These thresholds were used to binarize the image, and the result can be observed in Figure 4.2.1.b. From the figure, it is clear that the average colour model does not match the entire road area as there are many areas of the road that were not included after the thresholding operation. It can also be noted that additional areas from the foliage in the image did match the colour model and were included in the thresholded image. This is the reason why a convex hull algorithm was added to the road extraction algorithm.

The contours (Section 2.2.4.) within the image that match the average colour model are then determined from the thresholded image by using the standard `findContours` function. The function returns contour objects and for each contour object the size, exact positioning and hierarchical information is returned. The contours that were found for Figure 4.2.1(a) are shown in Figure 4.2.1(c). It is assumed that the largest contour in the image will include the road and therefore only this contour is sent to the convex hull algorithm.

The largest contour is received by the convex hull algorithm and the output of the algorithm is indicated in Figure 4.2.1(d). The green line indicates the original input contour. As was discussed in Section 2.2.5, the convex hull algorithm will determine all of the peaks (indicated by the green dots) and valleys (indicated by the red dots) found for the input contour. Finally, only the major outer peaks are determined and connected in a new contour which can be observed by the blue line in Figure 4.2.1(d). By extracting only this last contour from the image, the road is extracted, as is shown in Figure 4.2.1(e).

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

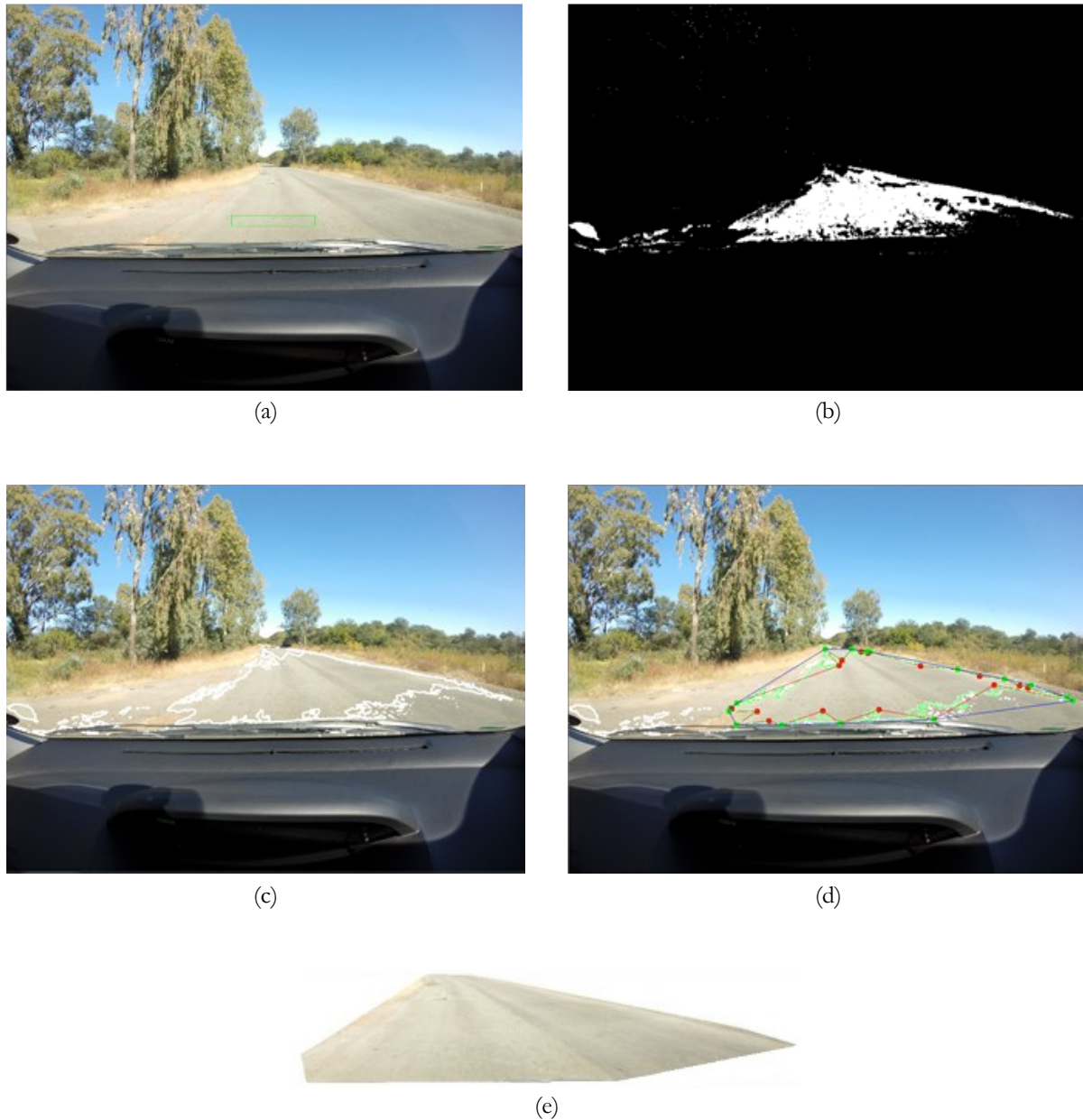


Figure 4.2.1 Road extraction algorithm example. In the figures (a-e) the inner working of the algorithm is demonstrated. In (a) the fixed area in the road is indicated by the green box. This is used to calculate the average colour model. By using the result and thresholding based on these values, (b) is obtained. The contours within the thresholded image is found and backprojected as in (c). Applying the convex hull algorithm and visually displaying its outputs results in (d). The result from extracting the convex hull algorithm is given in (e).

### 4.3 IMAGE PROCESSING APPROACH

The previous step applies to both the full image processing algorithm as well as to the simplified dataset and simplified classifier. The exact approach used exclusively in the image processing approach to detecting potholes is discussed here.



---

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

The output of the road extraction method is input to the pothole detection algorithm and is illustrated by the top two figures in Figure 4.3.1. The algorithm is applied to two examples with a less challenging example on the left and a more challenging example on the right. The example on the right side is considered challenging as it contains an oncoming vehicle and the road surface has deteriorated significantly. Each step of the algorithm is shown side-by-side. The next step is to convert the input image to greyscale (images second from the top) and to apply a Gaussian filter to the image in order to reduce the noise in the image. Recall from Section 2.2.3, that the Canny algorithm operates best when used in conjunction with a Gaussian filter. A Gaussian kernel of size  $3 \times 3$  was used because the average pothole height in the study is 17 pixels. The bigger the kernel, the more the image is blurred/smoothed which reduces the magnitude of the gradient change of the edges [91]. This is unwanted behaviour as the edges are crucial for pothole detection. Thus, there is a trade-off when performing smoothing.

The Canny edge detector was chosen as it provides good results for outdoor images by indicating the most important structures while avoid detecting the small details in texture [16] which for this investigation is desirable. The additional texture that can be detected from an asphalt road can interfere with the pothole detection and is therefore excluded. Another advantage of the Canny edge detector is that it has good performance when the algorithm is presented with high noise images [92]. The output of the Canny edge detector is shown in the figures in the middle of Figure 4.3.1.

The next step of the algorithm dilates the output of the Canny edge detector three times. Performing dilation on an image increases the number of the white pixels. An advantage of dilation in this setup is that the unwanted edges close to the outer boundaries, such as the wipers, become absorbed into the outer boundary leaving only the boundary contour visible. Another advantage of this approach is that it can aid the removal of other vehicles in the image and can be seen in the example in the right hand side of Figure 4.3.1. It is speculated that other vehicles on the road will normally be located close to the outer boundaries of an image and sufficient dilation would mean that the vehicles would also be absorbed into the outer boundary.

A final contour detection is then applied to the dilated image to find the potholes within the road section. The contours are filtered based on size constraints of the pothole model are discarded. This last step filters out any small defects in the road that are not classified as potholes as well as the larger contours found on the outer boundary of the extracted road. The tuning of the parameters is explained in Chapter 5.

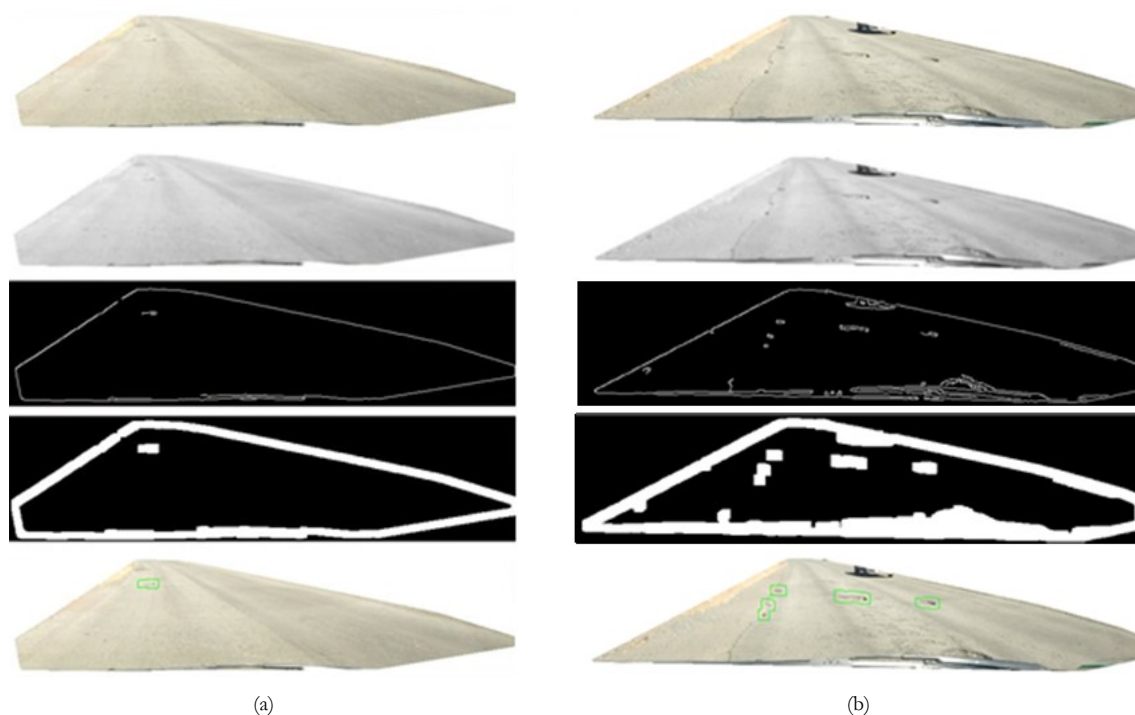


Figure 4.3.1 Pothole detection algorithm. A less challenging (a) and more challenging (b) example of the algorithm in action is given. First, the extracted road section is input to the algorithm which converts it to greyscale. The next step is to apply the Canny edge detector and can be observed in the middle two figures. The second last step dilates the output from the Canny edge detector. Contour detection is applied to the dilated image and by using sizing constrictions, the contours that are expected to be potholes can be determined. For illustration purposes, this result is backprojected onto the original input image and is displayed in the last two images at the bottom of the figure.

#### 4.4 DETECTOR WINDOW SIZE AND HOG DESCRIPTOR PARAMETERS

The selection of the detector window size for the machine learning algorithms is not a trivial matter as it is a major factor in the performance of the algorithm (both in processing time required and actual performance). Given the visual limitations of the camera, if small/potholes at a distance can be identified in the image, this setting will be the next minimum detection size limitation of the algorithm. This is because the size to which the input samples should be cropped and scaled are directly dependent on the detector window size. A detector window is used in both the cascade classifier, as well as the SVM machine learning algorithms used in this project.

No available guidelines was found on how best to select this window based on the particular object at hand. Therefore, a two-step approach was implemented to find the most

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

suitable window size. The first was to determine the distribution of the size of the potholes (Section 4.4.1) and in the second step it was decided to visualise the HOG features for candidate pothole samples (Section 4.4.2).

### 4.4.1 POTHOLE SIZE DISTRIBUTION

The size to which the potholes need to be normalised (Section 3.5) and the size of the detector window needs to be the same size. Therefore, the exact distribution of the potholes found in the data sets are determined. Histograms were created based on the pixel width and pixel height of the pothole distribution for both the simplified and complex scenarios are given in Figure 4.4.1 and Figure 4.4.2, respectively. All of the pothole width distribution histograms are on the left hand side of the figures and those on the right hand side indicate the pothole height distribution histograms. In both Figure 4.4.1 and Figure 4.4.2 the top histograms indicate the distribution of the positive training set and the bottom histograms indicate the positive test set. The window size was determined by *only* considering the training set. The test set distributions are given here to indicate that the pothole data in the test sets were selected in a fair manner and that the tests were not just performed on potentially more favourable data, for example, large and consequently more easily identifiable potholes). The size distributions of the pothole tests sets are nearly identical to that used in the pothole training set.

Note that all of these histograms were plotted by using the annotated data found on the downsampled images (1840 x 1380 pixels). By inspecting the histograms that pertain to the width of the potholes, the 30 – 39 pixel bin is seen to usually be the most populated. In all but one of the instances (Figure 4.4.2.d), the 10 – 19 pixel bin is most populated when inspecting the height information. From inspecting the histograms, the smallest reasonable detector window and normalisation size was selected as 32 x 16 pixels. This is a design choice and it is speculated that a smaller detector window would not contain enough pothole information that would negatively impact the performance of the classifier. The selected detector window size is investigated further in Section 4.4.2.

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

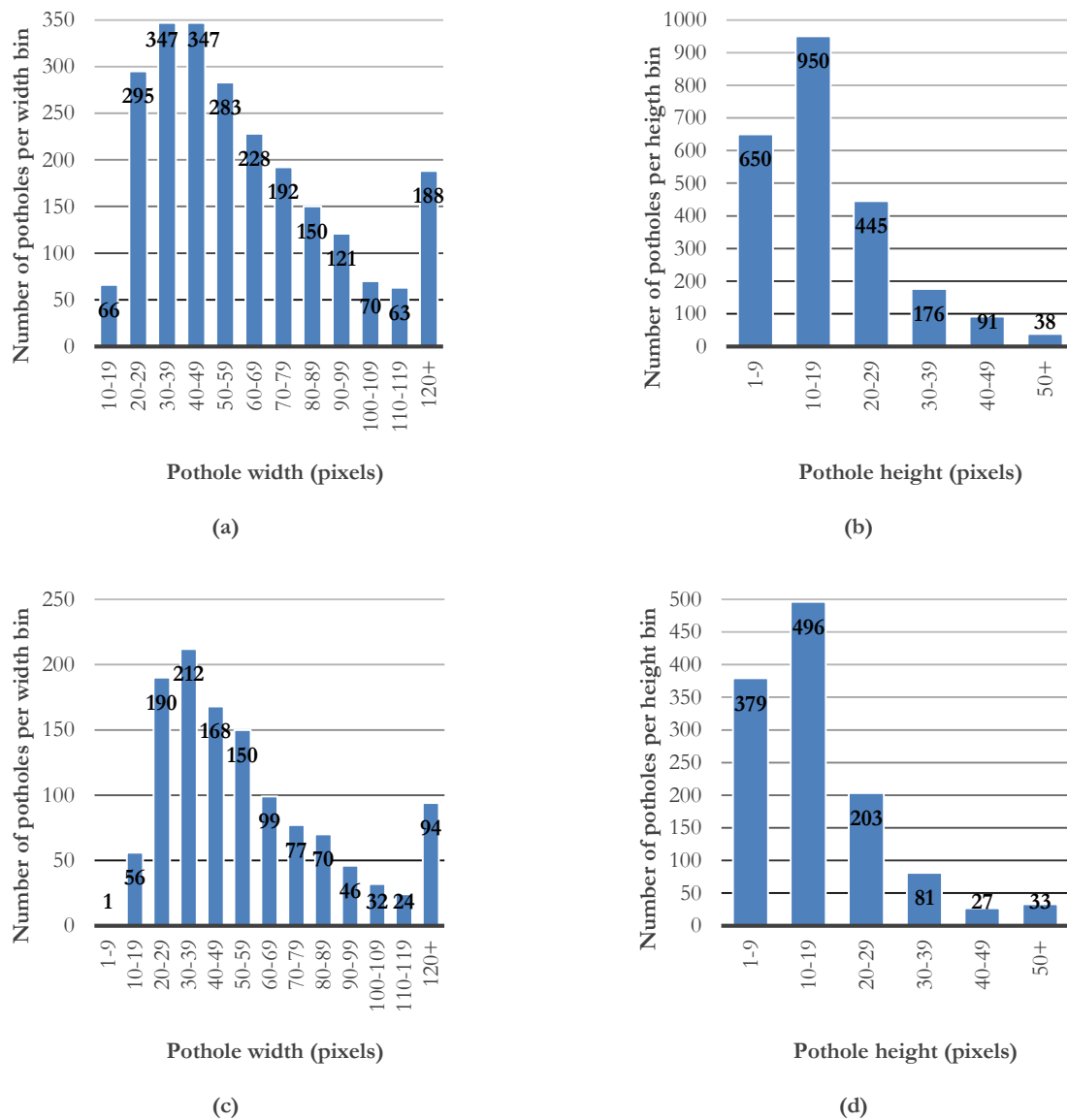


Figure 4.4.1 Simple scenario dataset pothole content breakdown. The top figures indicate pothole distribution for the training set and the bottom figures indicate the pothole distribution in the test set. Figures (a) and (c) show the pothole width distributions and figures (b) and (d) show the pothole height distributions.

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

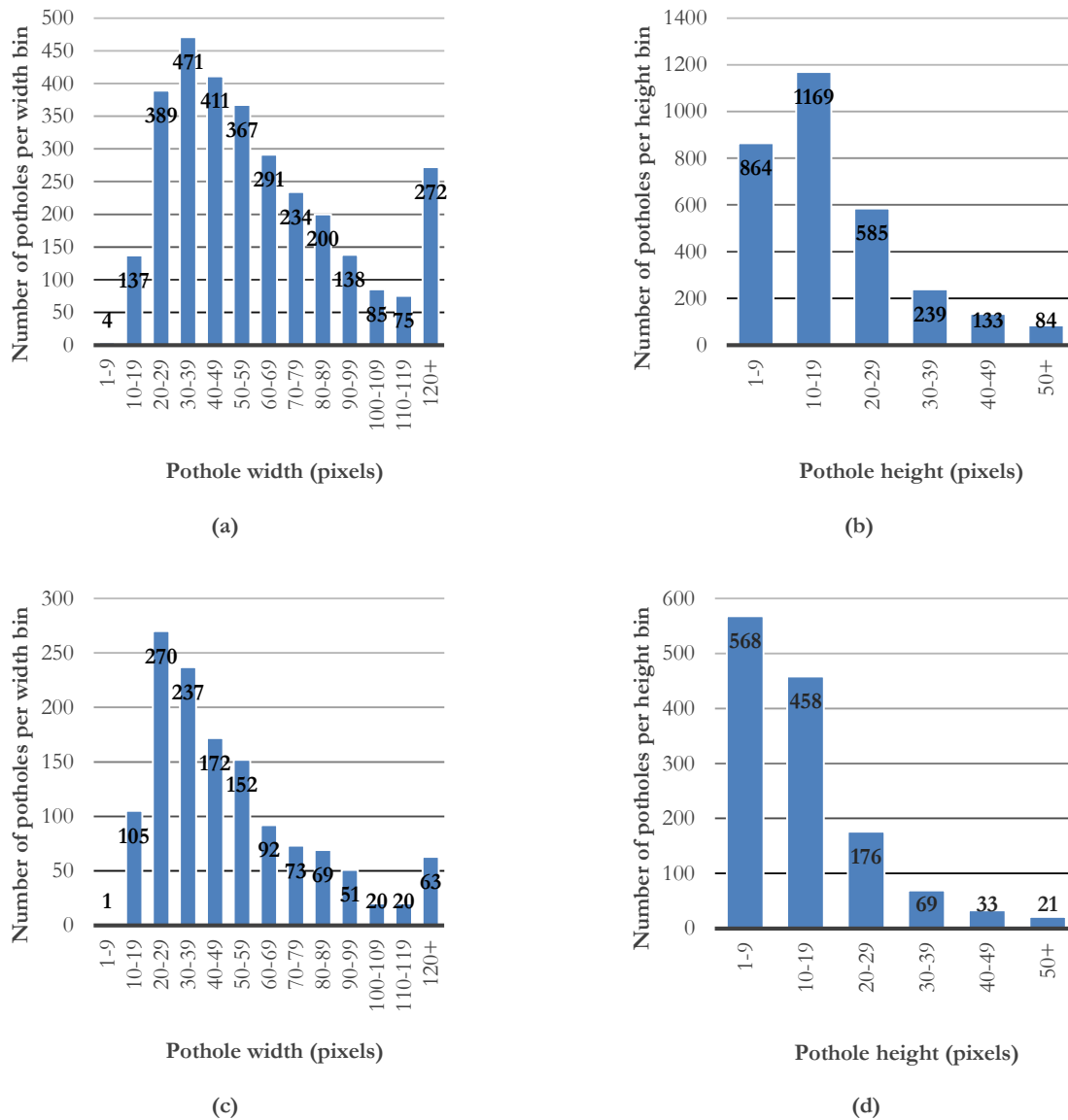


Figure 4.4.2 Complex scenario dataset pothole content breakdown. As in the previous figures, the top figures indicate pothole distribution for the training set and the bottom figures indicate the pothole distribution in the test set. Figures (a) and (c) show the pothole width distribution and figures (b) and (d) show the pothole height distribution.

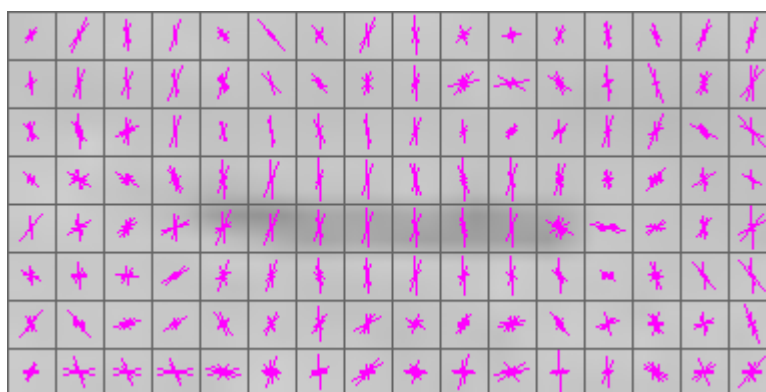
### 4.4.2 HOG DESCRIPTOR

In literature it was found that where a detector window was used in conjunction with HOG feature descriptors, both the window width and height were a multiple of a power of two [35, 93, 94].

The HOG descriptor can be visualised as is shown in Figure 4.4.3. An example 32 x 16 pixels greyscale pothole image is used for the illustration. The image is divided into a 16 x 8 grid

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

of cells (for a cell size of  $2 \times 2$  pixels) and for each of these cells, the HOG features are calculated as in Section 2.2.3. The purple lines in the image indicate both the magnitude and direction of the gradient. A longer purple line will indicate a higher gradient change while a shorter purple line indicates a smaller gradient change. The direction of the line indicates the direction of the gradient change. As is expected, the cells along the top and bottom borders of the pothole indicate strong gradient changes as the purple lines are long and vertical.



**Figure 4.4.3 HOG features visualisation.** An input image is divided into cells and for each cell the HOG descriptor is calculated. The purple lines in the figure indicate the HOG found in each cell. Both the direction and magnitude is expressed by each line. The length of the line indicates the magnitude while the orientation of the line indicates direction of the gradient change.

Note that as HOG features are used to identify shapes, it is necessary to include additional pixels around the pothole so as to include the road region's gradients and subsequent HOG feature details as well.

The HOG descriptor values used in this project are given in Table 4.4.1. In [34] and [35] it was found that the optimal number of orientation bins was nine. Increasing this number could possibly result in better performance but would increase the size of the feature vector. From the images it was found that a pothole is usually either a rectangular object that increases in height as it becomes closer or an elliptical object. As these shapes are fairly simple as compared to a five-point star for example, it is not necessary to increase the orientation bins. The corresponding orientation range of the orientation bins of the HOG feature with nine orientation bins is  $0^\circ - 180^\circ$ .

The default values for the HOG descriptor is optimised for people detection that is used in conjunction with a detector window of  $64 \times 128$  pixels. People are usually upright and therefore

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

the detector window used in people detection applications has a bigger pixel height than width. For potholes, it was found that the pixel width is bigger than the height. As the chosen detector window size in this project is one fourth the size of the default detector window, it was decided to scale down all of the HOG descriptor parameters by a factor of four. This maintains the same optimum ratio to calculate the features. Preliminary testing revealed that especially a smaller cell size (such as 2 x 2 pixels) was better suited to extract the HOG features from the potholes.

**Table 4.4.1 HOG descriptor parameter values.** The table indicates the chosen size of each parameter of the HOG descriptor. The cell size and stride size are chosen to be the same, while the block size is equal to 2 x 2 cells.

Parameter	Selected size
Cell size	2 x 2 pixels
Stride size	2 x 2 pixels
Block size	4 x 4 pixels
Window size	32 x 16 pixels
# Orientation bins	9 bins
Orientation range	0° - 180°
Normalisation	L2-Hys

The stride size is fixed at half of the block size and specifies how the blocks used to calculate the HOG features should overlap [35]. In this case, there is 50% overlap. The HOG descriptors for each cell are calculated for each stride of the grid. As described in Section 2.2.3, the HOG features need to be normalised over a particular grid (block size). Each block is normalised by the default L2-Hys method which is standardly used for the HOG descriptor [95, 96] and each feature is appended to the final feature vector. The final size of this feature vector per sample image is 3780 which is calculated by using equation (2.3-2).

To ensure that the results of the cascade classifier and SVM classifier are comparable, the detector window used for both has the same size.

## **4.5 SVM DEVELOPMENT**

Section 2.4.3 discussed the SVM and its corresponding kernels. A particular kernel must be selected to be used with the SVM. Each type of kernel has associated hyperparameters which is necessary to optimize the performance of the SVM. The procedure followed to obtain the hyperparameters are subsequently discussed in Section 4.5.1 and Section 4.5.2.

### **4.5.1 CROSS-VALIDATION**

In order to tune the parameters of a classifier, it is necessary to train the classifier with an initial selection for the parameters and then to test the performance of the classifier with these parameters. The process is repeated several times until the best values for each parameter has been determined. However, when using a data set that does not have an unlimited number of samples, overfitting can occur when training and testing the performance on the entire data set [97].

A good general practise to solve the problem of optimal parameter tuning is to divide the data into separate training, validation and test sets [14]. In this manner, the data can be trained on a separate training set, the parameters can be tuned on the validation set and the optimum values obtained from the validation set can be used to evaluate the true performance of the classifier on the test set.

For a dataset that is small, the data might not be enough to separate the data into separate training, validation and test sets. In these scenarios, a process called cross-validation is implemented which only separates the data into a training and test set. A variety of different methods can be used to determine how this split occurs, as well as the number of times the data should be split. Usually, the data is repeatedly split and the performance is measured for each split. The estimated performance of the classifier is determined by averaging the performance over all of the splits.

Two of the most used methods are k-fold cross-validation and LOO (Leave-one-out) cross validation [98].

In k-fold cross-validation, the data is split into k different sets and k-1 sets are used together as a training set and the k<sup>th</sup> set is used as the validation set and the performance of a particular set



---

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

of parameters are determined. The validation process occurs  $k$  times in such a manner that each of the  $k$  sets will be the validation set once. The value of  $k$  is selected by the user but the de facto standard has become a value of 10. The final performance is averaged over the  $k$  sets and this is deemed the performance of the classifier.

The LOO cross-validation method is very similar to  $k$ -fold cross-validation except that instead of dividing the set into  $k$  sets, the data is divided into  $n$  sets where  $n$  is the number of samples. Therefore, the validation process is repeated  $n$  times where  $n-1$  sets are grouped together and used for the training and a different single sample is left out and used for testing each time.

The only method that is implemented in OpenCV is the  $k$ -fold cross-validation and therefore this method is used in the project. In OpenCV the  $k$ -fold cross-validation is used only in conjunction with the grid search method (Section 4.5.2) to determine the hyperparameters for the SVM kernel [99]. This optimisation is done until the classifier evaluates to a user specified error or a specific number of iterations.

Conveniently,  $k$ -fold cross-validation does perform better than regular validation when the dataset is relatively small (in the order of a few thousand or less) [100].

### 4.5.2 GRID SEARCH

The manner in which the data set can be split was discussed in the previous section, however, the question remains, how the values used for the hyperparameters in the validation process should be selected and changed between different validation iterations in an optimum and efficient manner.

Dependent on the kernel used in conjunction with the SVM, there could be multiple hyperparameters that need to be tuned to develop the optimum classifier. For the RBF kernel there are two parameters, while in the polynomial kernel there are four. It is possible to determine the optimal values for the parameters manually via an empirical search, however, this is not desirable. In OpenCV, an automatic method to tune the hyperparameters for the kernels known as grid search is implemented [99]. To explain the grid search algorithm, an RBF kernel is used as an example. In the RBF kernel there are two hyperplane parameters namely,  $C$  and  $\gamma$ . Each parameter is varied individually in a logarithmic manner that is described by equation (4.5-1) [99].

---

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

$$\text{min\_val} * \text{step}^n < \text{max\_val} \quad (4.5-1)$$

In this equation the min\_val, step and max\_val parameters can be set and n is calculated from this information. Min\_val refers to the start value of the parameter, step is the step factor with which the parameter should change in between iterations and max\_val is the maximum value that the parameter must be. With this information, the number of steps (n) required to satisfy the equation can be calculated. In grid search, the size of the grid is equal to the product of the number of steps for each parameter. For example, if C and  $\gamma$  have four steps each, the size of the grid is 16. In [101], a strategy for finding the most suited values of the kernel via grid search is described. The approach first considers a coarse grid implying a wide range in between the start and end values for each parameter with a large step size. Once the optimal value is found in the coarse search, a finer search is implemented around the found optimal values by using a much smaller step size that will search in the vicinity of the optimal values found in the coarse grid.

In the project it was found that if the hyperparameters are known, the time to train the SVM is extremely quick (less than three minutes). However, when k-fold cross validation is used in conjunction with grid search, to obtain the optimum hyperplane parameters, the training procedure can take weeks dependant on how many parameters need to be optimised and how large the grid is. A formula for illustrating how many SVMs (s) need to be evaluated is given in equation (4.5-2) [102]. In the formula, c represents the number of classes of the classification problem. In terms of pothole detection, only potholes and non-potholes classes are considered which implies a bi-class classifier i.e.  $c = 2$ . The number of k-fold cross-validations is indicated by k. Each of the hyperparameters are represented by an individual  $n_i$ , where  $n_i$  is the number of steps for a particular hyperparameter in the grid. Finally, i represents the total number of hyperparameters.

$$s = c * k * n_1 * n_2 * \dots * n_i \quad \text{where } i > 0, i \in \mathbb{N} \quad (4.5-2)$$

If a polynomial kernel is being investigated in a bi-class SVM classifier with 10-fold cross-validation and each of the parameters are only varied in 4 steps, the number of SVMs that will have been evaluated to obtain the final optimised SVM is equal to 5120. If it requires roughly 3 minutes to train and evaluate each SVM, the time for this entire process to complete will be in the

---

## CHAPTER 4 – DEVELOPMENT AND METHODOLOGY

---

order of 11 days. Note, that this is the time for a very *rough* grid search. If a finer grid is used, the process will take much longer.

### 4.5.3 SVM KERNEL SELECTION

The only kernels that were considered in the project were the linear, RBF and polynomial kernels.

A linear kernel is a good selection if the data is almost linearly separable. When the number of features in the feature vector is very large, the linear kernel is generally a good kernel option [103]. As the kernel does not map the data to a higher dimension such as the other kernels, its computational cost is lower, therefore, its performance is faster.

The RBF kernel is also referred to as a Gaussian kernel. In preliminary tests, it was found that the  $\gamma$  value of the kernel for the complex scenario data set tended towards zero. This indicates that severe overfitting occurs [104]. Therefore, this kernel was disregarded.

The polynomial kernel was found to take the longest to complete the cross-validation and grid search process as the number of parameters are larger than that of the other kernels. A simple coarse grid search in a 10-fold cross-validation setup would take roughly 56 days to complete the training.

Considering all of the above information, it was decided to use a linear kernel for the SVM.

## 4.6 CONCLUSION

In this chapter key aspects of the design and methodology for both the image processing route as well as the SVM machine learning algorithm were discussed. Due to the nature of the implementation of the cascade classifier in OpenCV, no additional design process was required to obtain this type of classifier. Although the road extraction method is implemented using image processing algorithms, it is also used in the simplified scenario in conjunction with both machine learning algorithms. The crucial question of the of the detector window size was considered and it was concluded that a size of 32 x 16 pixels is appropriate.

## *CHAPTER 4 – DEVELOPMENT AND METHODOLOGY*

---

The validation process and hyperparameter selection process was also discussed. It was decided that a 10-fold cross-validation in conjunction with a grid search technique will be implemented to obtain the most suitable trained model. A coarse grid is firstly applied to identify the rough values that are suitable candidates and a finer grid search is implemented thereafter to find the most appropriate values for the hyperplane parameters. After a minor investigation into the various types of kernels that are available, it was decided to use a linear kernel for the SVM.

---

# CHAPTER 5

## RESULTS

### 5.1 INTRODUCTION

This chapter presents the key findings of the pothole detection investigation. The performance metrics that were used to analyse the classifiers are discussed in Section 5.2. The necessary settings of the OpenCV detectMultiScale function are given in Section 5.3. Then, from Section 5.4, the results are first presented per classifier, each in its own section. To allow better resolution on the curves, the maximum values of the axes were not kept constant across Sections 5.4-5.6. However, in Section 5.7, the results per scenario (simple or complex) are compared. An analysis of the performance of the classifiers based on the ranges at which they could detect potholes is presented in Section 5.8. In Section 5.9, each of the classifiers' performance is timed and discussed. Lastly, a conclusion to this chapter is presented in Section 5.10.

### 5.2 PERFORMANCE METRICS

There are various different methods to determine the performance of a classifier such as precision/recall curves [105] and ROC (Receiver Operating Characteristic) curves [106].

The selection of the particular curve depends on the nature of the detection problem. For pothole detection, it was only necessary to classify a region as a pothole (belongs to the positive class) or as a non-pothole region (belongs to the negative class). Given the size of the potholes and the selected detector window, it was found that the number of non-pothole regions outnumbered the number of pothole regions for a given image. Also, by combining the detector/sliding window with an image pyramid (Section 2.4.4) to improve the scale-invariance of the detector, the number of non-pothole regions increase exponentially compared to the number of pothole regions. Therefore, thousands of negative samples are generated and only a few positive samples per image. When the ratio between the two classes are very large/small and not roughly equal, the problem is defined as unbalanced [107].

## CHAPTER 5 – RESULTS

---

A precision/recall curve does not take the true negatives into account, however, a ROC curve does. In an unbalanced problem, an ROC curve will not indicate the performance with respect to the smaller class with sufficient resolution due to the imbalance [108, 109]. Therefore, it was decided to use precision/recall curves to measure the performance of the classifiers. The formulas used to calculate the precision and recall is given in equations (5.2-1) and (5.2-2), respectively. Equation (5.2-1) indicates the ratio between the number of correctly detected potholes (true positives) to the number of falsely detected potholes (false positives). The number of the true positives and false positives are not necessarily dependent on each other and can vary independently. Equation (5.2-2) indicates the ratio of the number of correctly detected potholes to the number of missed potholes (false negatives). The total number of potholes in the image is equal to the number of true positives plus the number of false negatives and therefore, these two numbers are dependent on each other. As the number of true positives increase, the number of false negatives has to decrease and vice versa.

By selecting a variable of the classifier and varying it over a range, the output precision and recall per instance of the variable can be recorded and plotted.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \quad (5.2-1)$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \quad (5.2-2)$$

In order to determine the optimum point for the particular settings of a classifier, it was decided to use the F1 score indicated in equation (5.2-3) which is a special case of  $F_\beta$  score [110] where  $\beta$  was selected to be equal to one. This value for  $\beta$  implies that misclassifying negative samples as positive samples, is regarded as equally detrimental to missing potholes and thereby increasing the false negatives in the system.

$$\text{F1 score} = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.2-3)$$

The performance was determined with OpenCV's performance.exe utility. It requires an input file containing the correct locations of the potholes in rectangle form and compares this to the output rectangles of a classifier. According to the PASCAL VOC challenge [111], an object is

## CHAPTER 5 – RESULTS

---

deemed correctly detected when a particular output rectangle from the classifier covers at least 50% of the known object location. Another requirement of the contest is that multiple detections of the same object be penalised and count towards the false positive total. Therefore, the performance.exe utility's settings were kept to their original settings as these settings meet the requirements of the PASCAL contest.

### 5.3 MACHINE LEARNING PROGRAM SETTINGS

The functions and settings used in OpenCV to train the cascade classifier is given in Appendix A.

Both the cascade classifier and HOG descriptor have an implementation to perform a window scan, as well as the image-scaling and is known as `detectMultiScale` [112]. The respective `detectMultiScale` functions have a few similarities in their implementation for the cascade and HOG descriptors, however, there are some differences. The details of this function and its parameters are given in Appendix B.

When image-scaling takes place, the possibility exists that an object could be detected on more than one scale and therefore leads to multiple detections. The scaling is controlled by a parameter called `scaleFactor` (cascade classifier) or `scale0` (HOG descriptor). This setting was selected to be 1.1 and therefore implies that for each step in the image pyramid, the image is reduced in size with 10%. An additional parameter of the `detectMultiScale` function attempts to limit the multiple detections and is either known as `minNeighbors` (cascade classifier) or `group_Threshold` (HOG descriptor). An illustration of the effect of this parameter for the SVM+HOG classifier in the simple scenario is given in Figure 5.3.1. Note that a similar output will be achieved by using the LBP cascade classifier, although, the results might not be the same. In the top image in Figure 5.3.1, `group_Threshold` is set to zero and therefore, no grouping takes place. Therefore, there are many detections of the same pothole, all except one will count towards the false positives of the classifier. By adjusting the `group_Threshold` parameter appropriately, the bottom image in Figure 5.3.1 can be obtained. In this image it can be seen, that all of the multiple detections within the same region, are grouped together to form one detection. However, if `minNeighbors` is set to five groupings, for example, and there are six detections, only five rectangles will be included in one detection leaving one false detection. Similarly, if there

## CHAPTER 5 – RESULTS

---

are four detections, there are not enough detections to satisfy the parameter and no detections will be output, leading a false negative.



Figure 5.3.1 Example of the effect of grouping detections. The images above indicate the effect of not grouping (top image) or by grouping (bottom image) detections in the same region.

The SVM+HOG classifier has an additional parameter setting for the `detectMultiScale` function and is known as `hit_threshold`. This parameter allows the user to adjust a threshold between the classifying plane and the obtained features after training occurred which will adjust where the separating line is that classifies samples as positives or negatives.

### 5.4 RESULTS OF IMAGE PROCESSING

The precision/recall curves shown in Figure 5.4.1 and Figure 5.4.2 show the results obtained for the image processing algorithm developed in Section 4.3. The Canny function's setting was chosen as the parameter to obtain the curve by varying it from 10 to 100. The curves start off at roughly zero precision and recall when the Canny setting is 10 and are traced as the Canny setting is increased in steps of 5 until the maximum allowed value (100). When the Canny setting is low, it detects only the strongest edges and as it is varied to 100 it detects more subtle edges. The Canny setting was varied for three different dilation settings namely 4, 5 and 6. A higher



## CHAPTER 5 – RESULTS

dilation setting results in larger white pixel regions (produced by the Canny function) than a lower dilation setting would produce.

The curves have an increasing trend as the Canny setting is increased. However, at a certain setting for the Canny function, the precision (roughly) stabilises and the recall decreases. Therefore after this point, the ratio of the correctly detected potholes to false detections remain the same, however, the decreasing recall indicates that fewer and fewer potholes are being detected by the algorithm. For an increasing Canny setting, the edge detector becomes more and more relaxed with its thresholds and more edges (especially those close to each other or on the boundary of the extracted road) are detected in the image. Due to the dilation function, many of these edges are absorbed together into one contour and could result in the detected potholes becoming too large and they are therefore discarded by the algorithm.

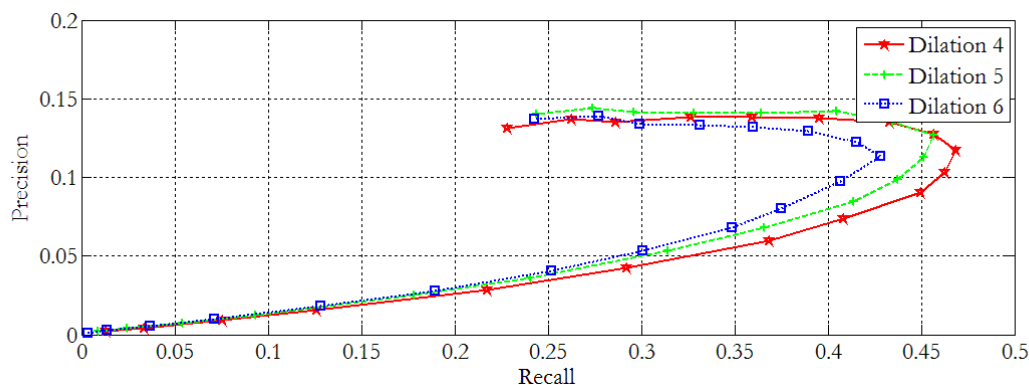


Figure 5.4.1 Image processing simple scenario precision/recall curves. By varying the setting of the Canny function (from 10 to 100) and keeping the dilation constant (per curve), it was possible to produce the curves presented. The best results were obtained at a dilation of 5.

Graphs similar to those obtained for the simple scenario can be observed for the complex scenario. However, in the complex scenario, the precision is in general much lower. As there is more foliage present in the complex scenario images, the number of false detections relative to the detected potholes is very high. The recall of the curves in Figure 5.4.2 is very low which implies that many of the potholes in an image are not detected. This is because when the lighting conditions in the area of the road vary greatly, the road extraction algorithm often fails. This failure can be observed in a number of ways such as only extracting the dash board of the vehicle or a very small portion of the road. Naturally, if there are no potholes present in the extracted region

## CHAPTER 5 – RESULTS

that is sent to the rest of the algorithm, it is not possible to detect the potholes present in that particular image, which increases the number of false negatives.

The optimal settings for each scenario was found by determining the highest F1 score for each scenario. In the simplified scenario, it was found that a dilation of 5 and a Canny setting of 75 yielded the highest F1 score (0.210729). The highest F1 score in the complex scenario is 0.06033 and is achieved by setting the dilation to 5 and the Canny edge detector to 70. Note that in both scenarios, the values for the settings are almost equal. The F1 scores of the different scenarios should not be compared to each other, as both scenarios differ vastly in their deployment scenario. However, the different algorithms *per scenario* are comparable and this comparison is given in Section 5.7.

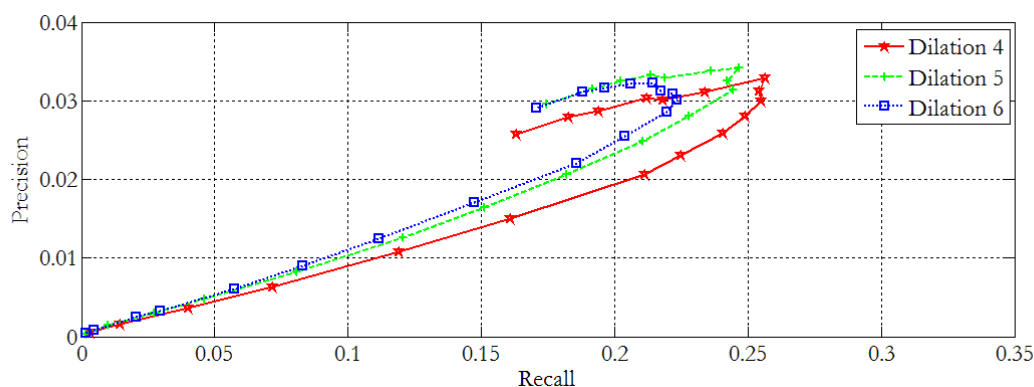


Figure 5.4.2 Image processing complex scenario precision/recall curves. As in the simple scenario, by varying the setting of the Canny function from 10 to 100 a curve can be obtained per dilation setting.

## 5.5 RESULTS OF CASCADE CLASSIFIER

This section presents the results of the LBP cascade classifier. The LBP classifiers were each trained to the maximum number of stages that they could achieve given the number of samples and the small size of the detector window. For the simple LBP classifier, it was found that the maximum number of stages to which the classifier could be trained was 25, while in the complex LBP classifier, the maximum number of stages that the classifier could be trained to was 21. The maximum number of stages that can be trained to in OpenCV is limited by the size and difficulty of the positive sample set [113]. As training for the highest number of stages does not guarantee the best results, the results of other LBP cascade classifiers trained to fewer stages are also presented.

## CHAPTER 5 – RESULTS

The precision/recall curve given in Figure 5.5.1 presents the results of the LBP classifier for the simple scenario while Figure 5.5.2 presents the results of the LBP classifier for the complex scenario. All of the curves were obtained by varying the `minNeighbors` parameter of the `detectMultiScale` function between 0 and 9 for a classifier trained to a particular stage. From Section 2.2.4 it is clear that a region in an image that has a positive detection, might have other positive detections on the images created in the image pyramid, but in the same region. The `minNeighbors` parameter specifies the amount of detections across an image pyramid required for the classifier to output a positive detection in the region. If less detections are present, the output for the classifier in that region is discarded and if more detections are present, multiple detections in the same region is given as the output.

It was observed that in both Figure 5.5.1 and Figure 5.5.2 the precision decreases and the recall increases as `minNeighbors` is varied from 9 to 0. Therefore, when a larger number of detections are grouped together, the precision increases while the recall decreases. This implies that regions where there are not potholes in the image, do not have many detections in the image pyramid and these false positives can be removed with this variable. However, the decrease in the recall indicates that the pothole regions in the images do not produce enough multiple detections in the image pyramid and are subsequently removed.

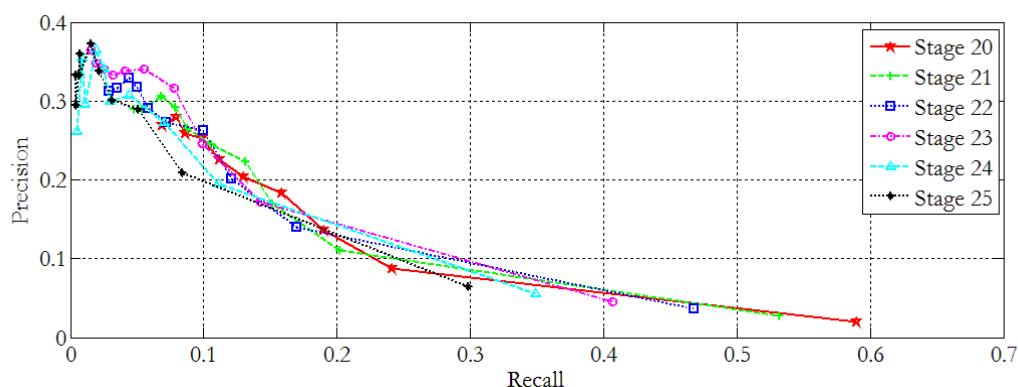


Figure 5.5.1 LBP simple scenario classifier precision/recall curves. Various stage classifiers were tested ranging from 20 stages to 25 stages) and the curves were obtained by varying the `minNeighbors` variable from 0-9.

In Figure 5.5.1 and Figure 5.5.2, it can be seen that for a given `minNeighbors` value the recall decreases as the number of stages for the classifier increases. Each stage of a classifier calculates different features and from the recall results it is implied that the features at the higher stages are not properly suited to detect potholes. Therefore, the higher stages rejects potholes based on these features which results in a higher number of false negatives. The precision results

## CHAPTER 5 – RESULTS

at higher values for the recall (greater than 0.3 for the simple scenario and greater than 0.07 for the complex scenario) are relatively similar in value per scenario type for the classifiers. Therefore, when a smaller number of detections across the image pyramid are grouped together, the number of false positives detected for each of the classifiers does not change significantly as the number of stages increases. In general, it can be seen that the precision/recall curve results does not change dramatically as the number of stages are varied.

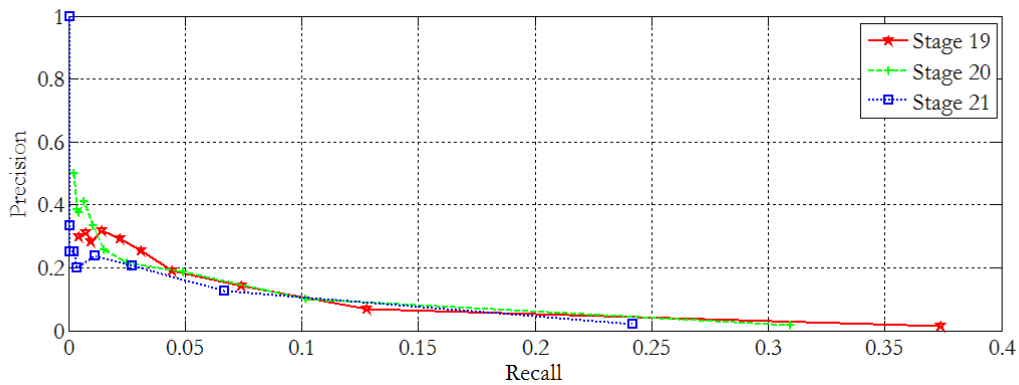


Figure 5.5.2 LBP complex scenario classifier precision/recall curves. In this curve the results are shown for classifiers that consist of 19, 20 and 21 stages. The curves were obtained by varying the `minNeighbors` variable from 0-9.

The highest F1 score found in the simplified scenario was 0.169969 for a 20-stage classifier and a `minNeighbors` setting of 3. A 16-stage classifier and a `minNeighbors` setting of 6 yielded the highest F1 score (0.129457) in the complex scenario.

## 5.6 RESULTS OF SVM

Figure 5.6.1 present the simple scenario results and in Figure 5.6.2 the results of the complex scenario is given. The `hit_threshold` parameter adjusts the distance between the decision boundary and the features, and the `group_threshold` parameter is similar to that of the cascade classifier [114]. The `hit_threshold` parameter was varied for each curve while the `group_threshold` parameter was kept constant per curve. The `hit_threshold` parameter was varied from 0.0 to roughly 2.7 in 0.1 intervals dependent on which value of the `hit_threshold` for a given classifier results in a null output (0% precision and 0% recall). The

## CHAPTER 5 – RESULTS

null output was found when the `hit_threshold` was at its highest value. The curve is traced by lowering the `hit_threshold` from its maximum value found to 0.0.

In both scenarios, the recall increases and the precision decreases as the value of the `hit_threshold` is decreased from its maximum value. At the apex on the recall curve, the recall decreases while the precision continues to decrease for lower values of the `hit_threshold`. Therefore, for higher values of the `hit_threshold`, the area included by the positive decision boundary increases which allows more input samples to be classified as positive. This results in a decrease in the number of missed potholes and, unfortunately, also an increase in the number of false positives. However, after the recall apex, the `hit_threshold` is low enough that the recall decreases while the precision continues to decrease. The smaller positive decision area results in a higher number of false negatives, as more potholes are not included in the positive decision area. This causes a decreasing recall for the lower values of `hit_threshold`.

In terms of the `group_threshold`, the results indicate that there isn't a significant difference when this parameter is varied for the indicated ranges and that the performance of the different classifiers per scenario remain roughly the same.

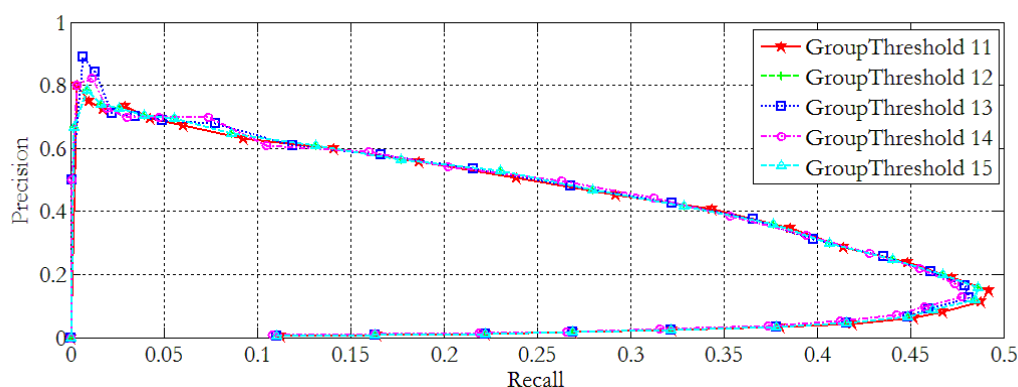


Figure 5.6.1 SVM+HOG simple scenario classifier precision/recall curves. The results that were obtained by varying the `hit_threshold` parameter from 0.0 in intervals of 0.1 until the classifier could no longer yield a result and stopped at zero.

## CHAPTER 5 – RESULTS

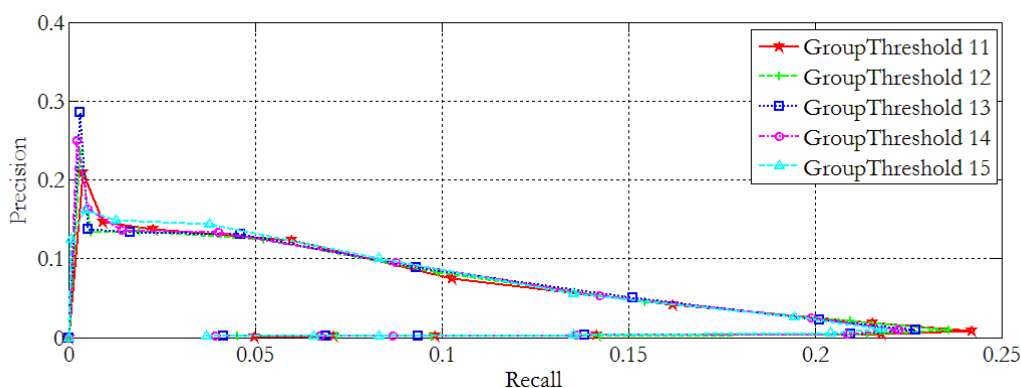


Figure 5.6.2 SVM+HOG complex scenario classifier precision/recall curves The results that were obtained by varying the hit\_threshold parameter from 0.0 in intervals of 0.1 until the classifier could no longer yield a result and stopped at zero.

Using the F1 score metric, the highest score for the simple scenario classifier is 0.372942 for a group\_threshold of 11 and a hit\_threshold of 1.5. The maximum F1 score for the complex scenario is 0.091088. This score is found at a group\_threshold setting of 14 and a hit\_threshold of 0.8.

### 5.7 COMPARISON OF ALL CLASSIFIERS

The curves on which the maximum F1 score was found for each of the classifier/scenario pairs of the previous sections are presented in this section. A table containing all of the exact values at which the predicted optimum F1-score occurs per classifier is given at the end of the section in Table 5.7.1.

The maximum F1 score curves for the simple scenario are shown in Figure 5.7.1. In the figure, it is clear that the SVM+HOG classifier outperforms both the LBP cascade classifier and the image processing algorithm. This difference in results could be attributed to the HOG features that focus on shapes and are more suitable for an object with a higher variety in shape. Based on the maximum F1 score, the second best performing classifier is the image processing algorithm. For the image processing algorithm, the shape of the detection is not relevant and therefore it detects more potholes than the LBP cascade classifier when it is tuned to its maximum F1 score. However, for this same reason, the image processing will detect any shape in the road as a pothole which results in more false positives. Therefore, the image processing algorithm's precision is

## CHAPTER 5 – RESULTS

lower than that of the SVM+HOG classifier. The maximum F1 score point of the LBP cascade classifier is the lowest of the classifiers in the simple scenario. For this metric, it is suggested that LBP features are not suitable for detecting potholes and is possibly do to LBP features being more suited to detecting objects with invariant shape properties.

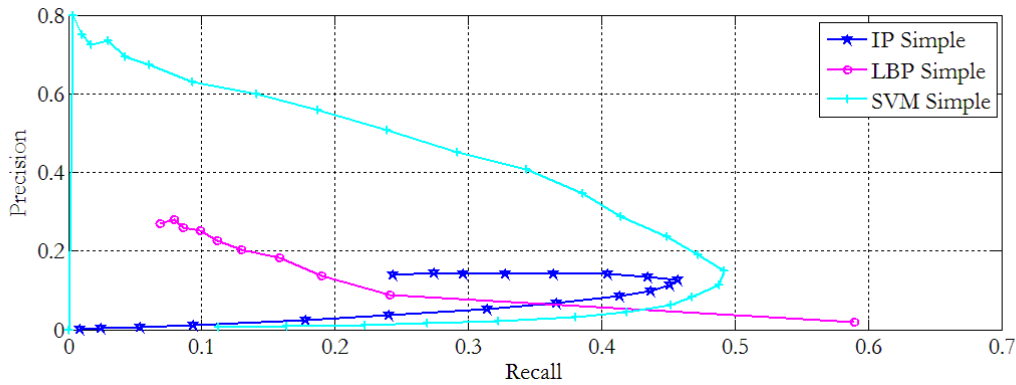


Figure 5.7.1 Maximum F1-score classifiers for the simple scenario. The best image processing algorithm, LBP cascade classifier and SVM+HOG classifier curves are presented.

The complex scenario's maximum F1 score curves per classifier were determined for the image processing algorithm, LBP cascade classifier and SVM+HOG classifier and is given in Figure 5.7.2.

The maximum F1 score for the complex scenario is found on the LBP cascade curve. Section 2.3.1 discussed the properties of the LBP cascade classifier and it was found that if the lighting in an image is varied uniformly, the LBP features can still recover the features (Figure 2.3.2). This could be the reason why the LBP cascade classifier outperforms the image processing algorithm and SVM+HOG classifier, as it can detect more potholes that are covered by shaded regions.

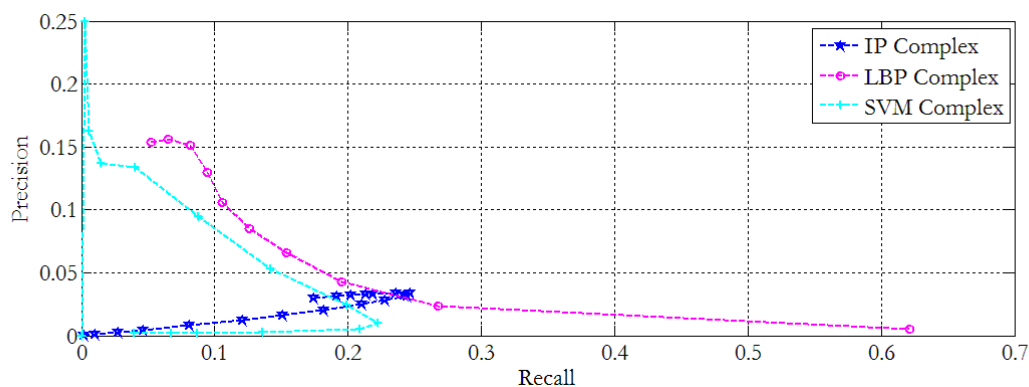


Figure 5.7.2 Maximum F1-score classifiers for the complex scenario. The best image processing algorithm, LBP cascade classifier and SVM+HOG classifier curves are presented.

## CHAPTER 5 – RESULTS

The SVM+HOG classifier's maximum F1 score is close to that of the LBP cascade classifier. Therefore, it is suspected that the HOG features are relatively capable of extracting pothole regions in mixed lighting conditions and can give comparable results to LBP features. The image processing algorithm performed the worst based on the F1 score metric. As was stated in Section 5.4, the performance of the image processing algorithm is directly related to how well the road section could be extracted. If the road is poorly extracted and does not include the pothole regions, the number of false negatives will be increased, and therefore, the recall will be lower. This is, however, not the case. It can be seen that for the image processing algorithm, the precision is, in general, much lower than for the other complex scenario classifiers. Therefore, the ratio of false detections to detected potholes is much higher. In the mixed lighting conditions, there are many regions in the road that have patches of light due to, for example, tree canopies (see the bottom figure in Figure 1.1.1). The image processing algorithm only detects edges and does not include the direction of the gradients (as with HOG features). Therefore, the number of false positives are increased, leading to a lower precision.

**Table 5.7.1 Optimal classifier settings and performance based on F1-score.** In the table, each of the classifiers that had the highest F1-score in their particular scenario is given with the exact parameters that achieved the highest F1-score.

Classifier type and scenario	Details	Highest F1-score
Image processing (simple)	Dilation 5 (Canny 75)	0.210729
LBP cascade (simple)	Stage 20 (minNeighbors 3)	0.169969
SVM+HOG (simple)	GT 11 (HT 1.5)	0.372942
Image processing (complex)	Dilation 5 (Canny 70)	0.060033
LBP cascade (complex)	Stage 16 (minNeighbors 6)	0.109457
SVM+HOG (complex)	GT 14 (HT 0.8)	0.091088

Images indicating the pothole detection by using the developed image processing algorithms can be found in Figure 4.3.1. To demonstrate the pothole detection for the machine learning algorithms, the SVM+HOG classifier was used at its highest F1 score settings (Table



## CHAPTER 5 – RESULTS

5.7.1) was used. Note that the LBP cascade classifier will have a similar output (green rectangles), however, the LBP cascade classifier would not necessarily have the same results. An example image indicating the output for the simple scenario is given in Figure 5.7.3. In this example there were four potholes of which the algorithm could detect three and had one false positive detection due to the shadow of a tree.

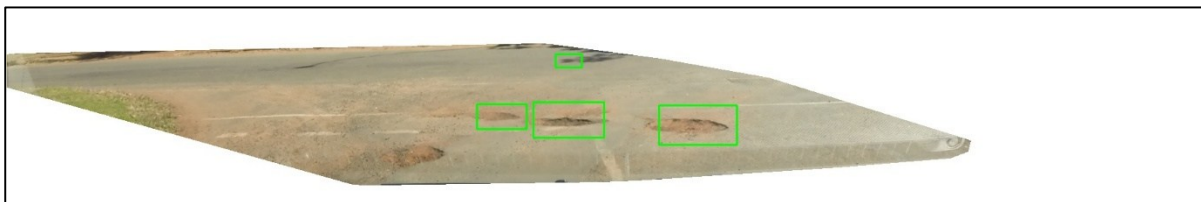


Figure 5.7.3 Example output of machine learning classifier in simple scenario. The figure shows an example of the output obtained from the SVM+HOG classifier. Note that a similar type of output (green rectangles) will be obtained by using the LBP cascade classifier.

An example of the output in the complex scenario is given in Figure 5.7.4. Again, the SVM+HOG classifier was used as the example for the machine learning algorithms. In this image there was only one pothole and was accurately detected, however, there was also one false positive detection in a tree.

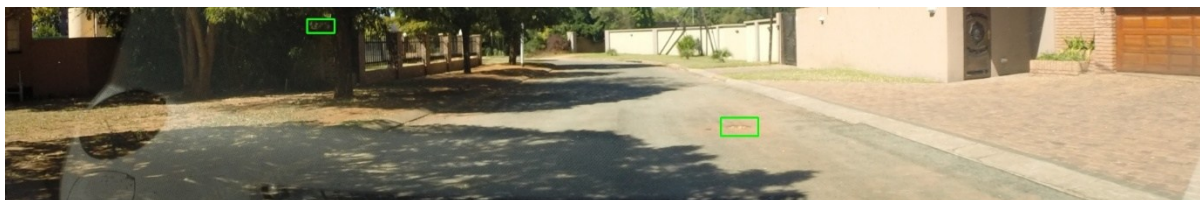


Figure 5.7.4 Example output of machine learning classifier in complex scenario. The figure shows an example of the output obtained from the SVM+HOG classifier. Note that a similar type of output (green rectangles) will be obtained by using the LBP cascade classifier.

## 5.8 ANALYSIS OF RESULTS WITH RESPECT TO DEPTH

Each point on each of the precision/recall curves found in the previous sections can be further evaluated based on the depths of the potholes that could be detected. Since there are roughly 1000 such points, only the detected potholes at the maximum F1 score is evaluated per classifier.

## CHAPTER 5 – RESULTS

---

There are, however, two practical considerations for this section. First, it is important to correct the fisheye effect (Section 3.7) on the pixel where the pothole occurs. The method for correcting the fisheye effect on a pixel is given in Appendix C. Second, the results presented here are given relative to the position that the camera was mounted to the vehicle. Therefore, the additional distance between the camera and the grill of the vehicle (+/- 1 m) should be subtracted from the results if the true distance from the *vehicle* is desired.

From the investigation in Section 3.7, the method that was determined to be the most accurate for depth estimation in monocular images was the cross-ratio formula. Therefore this formula is applied here to determine the distance at which the potholes occur from the *camera*.

In Figure 5.8.1, the histograms that are obtained for the simple scenario are given. Note that when, for example, a range indicates 0 – 4 m, it implies all of the potholes in the region of 0 to 4.99 m in front of the camera. In Figure 5.8.1(a), the distribution of the total available potholes in the simple scenario is given. Figure 5.8.1(b), indicates the distribution of the detected potholes when using the image processing technique, while Figure 5.8.1(c) indicates the detected pothole distribution for the LBP cascade classifier. Lastly, Figure 5.8.1(d) presents the detected pothole distribution found when implementing the SVM+HOG classifier.

When considering the number of potholes that were detected, it can be seen that for all of the various ranges, the image processing algorithm detected the highest number of potholes. From the images, it was observed that potholes that are far away from the camera have less visible detail than those up close. Therefore, it was expected that all of the algorithms would detect a lower percentage of potholes. This is due to the fact that less features are evident and the potholes are virtually reduced to slightly grey horizontal lines that are only a few pixels in height. The general trend lines in Figure 5.8.1(b), Figure 5.8.1(c) and Figure 5.8.1(d), indicated that each of the classifiers' results followed the distribution in the data (Figure 5.8.1(a)), except at the 30+ m range. At 30+ m it can clearly be seen that the algorithms struggled to detect any potholes.

The depth estimation results for the complex scenario are given in Figure 5.8.2. As in the simple scenario, the distribution of the detected potholes per range is similar to that of the actual pothole distribution in the scenario Figure 5.8.2(a).

The 30+ m range is important as it indicates the current system's vehicle's maximum speed. Therefore, if a vehicle travels at 120 km/h (roughly 33 m/s), regardless of the additional blur

## CHAPTER 5 – RESULTS

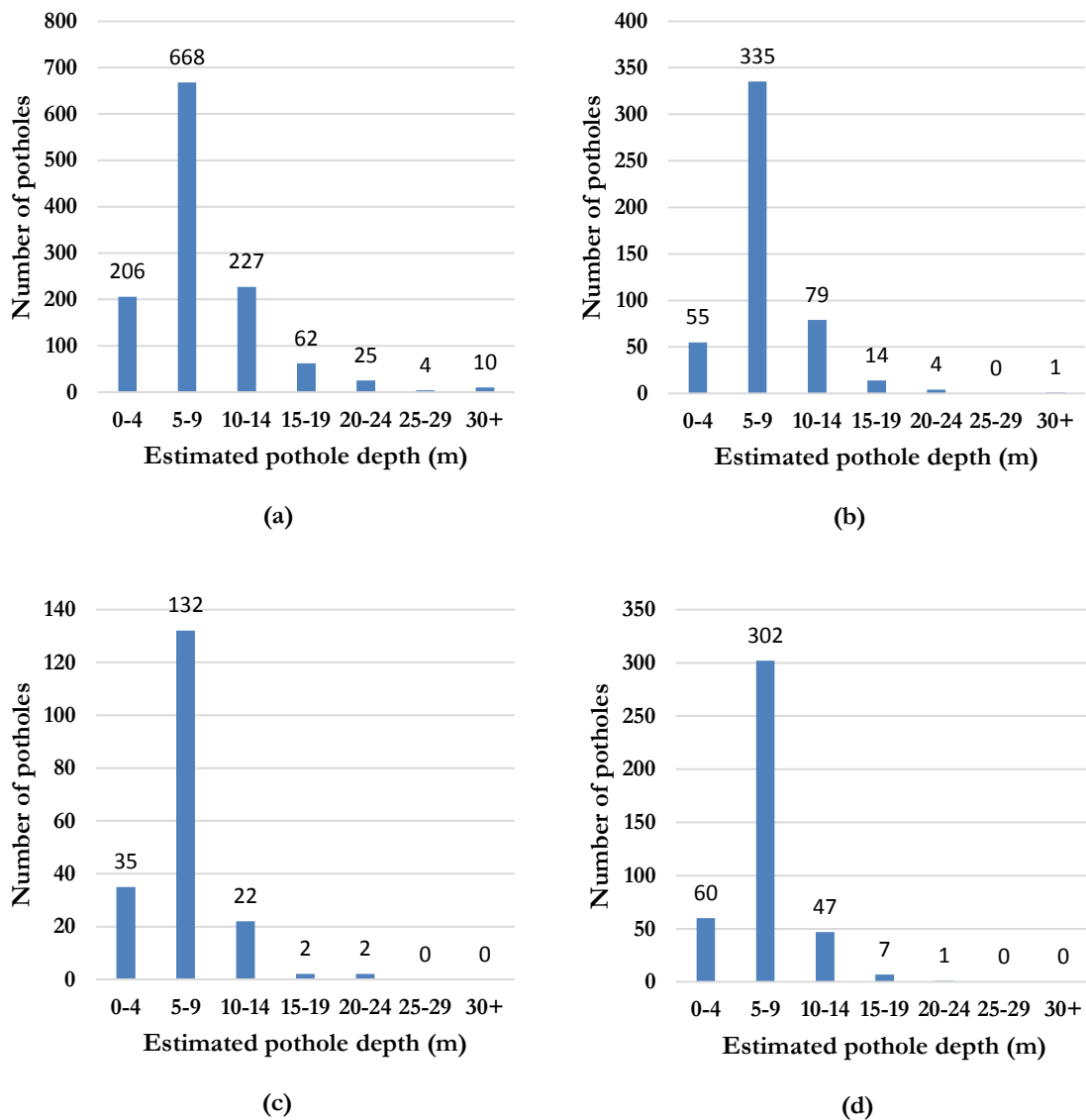


Figure 5.8.1 Simple depth estimation results. In (a) the actual distribution of the potholes in the simple scenario is given. The pothole distribution that could be in the simple scenario by the image processing, LBP cascade classifier and SVM+HOG classifier are indicated in (b), (c) and (d) respectively.

present in the image at this speed, it can be seen that potholes in this system cannot be detected far enough ahead for detection purposes.

To simplify the findings, the potholes that could be detected for the various ranges can be expressed as a percentage of the total number of potholes available in the scenario. These findings are given in Table 5.8.1.

## CHAPTER 5 – RESULTS

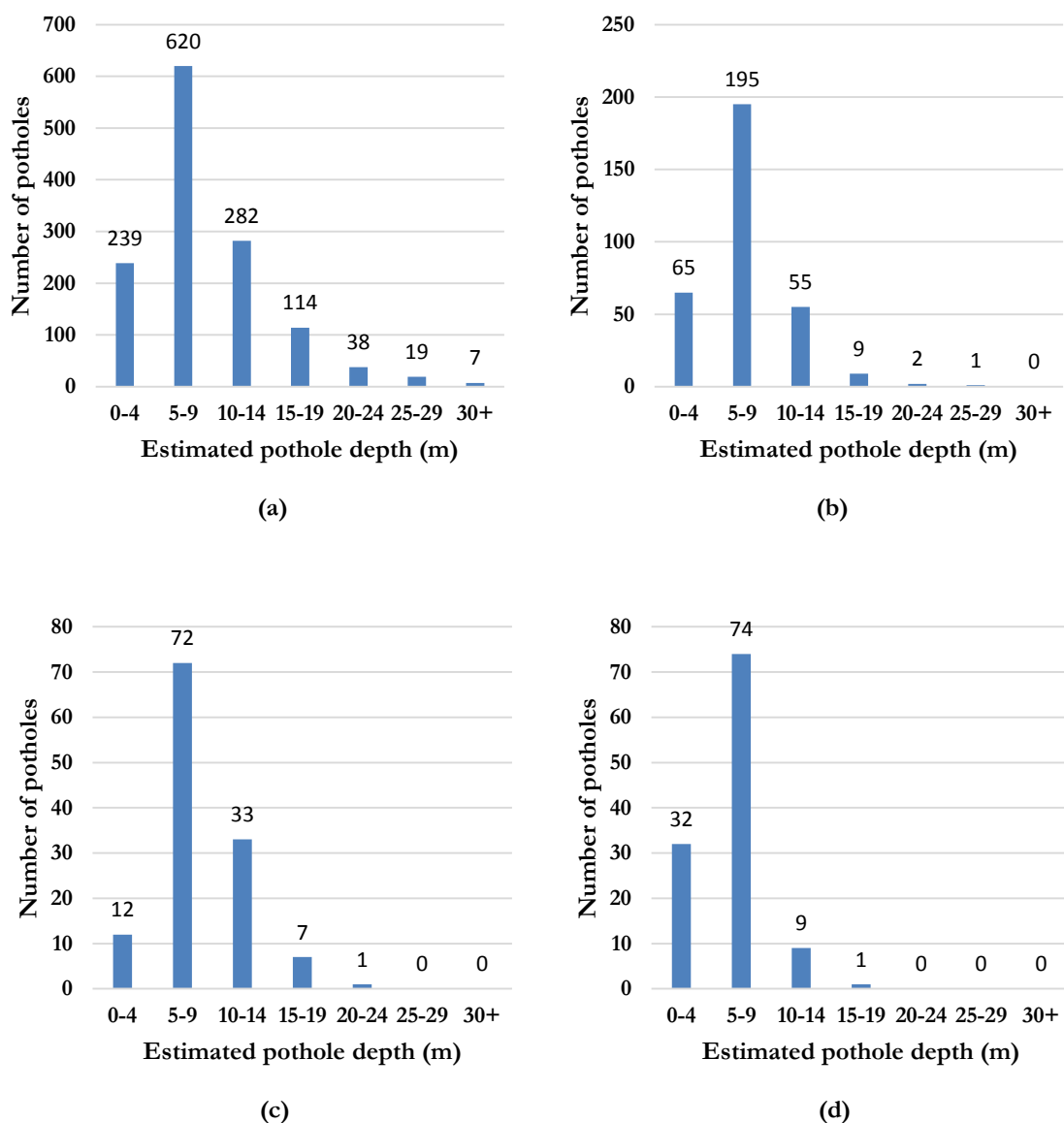


Figure 5.8.2 Complex depth estimation results. In (a) the actual distribution of the potholes in the complex scenario is given. The pothole distribution that could be in the complex scenario by the image processing, LBP cascade classifier and SVM+HOG classifier are indicated in (b), (c) and (d) respectively.

For all of the classifiers, except for the SVM+HOG classifier in the complex scenario, it was found that the percentage of potholes that could be detected in the range 0-4 m was lower than in the 5-9 m range. This can be caused by several different factors. In the case of the image processing, potholes that are close to the outer boundary of the extracted road can be absorbed into the boundary due to the dilation function. Considering the machine learning classifiers, it must be remembered that each pothole is detected multiple times due to the scaling of the image to detect potholes at different ranges.

## CHAPTER 5 – RESULTS

**Table 5.8.1** Detected potholes expressed as a percentage as a function of depth. The percentage of potholes that each classifier could detect for its own scenario is indicated in the table. In general, as the potholes were located further away from the camera/vehicle, the lower the number of detected potholes.

Depth (m)	Simple scenario potholes detected %			Complex scenario potholes detected %		
	Image processing	LBP cascade classifier	SVM+HOG classifier	Image processing	LBP cascade classifier	SVM+HOG classifier
0-4	26.70	16.99	29.13	27.20	5.02	13.39
5-9	50.15	19.76	45.22	31.45	11.61	11.94
10-14	34.80	9.69	20.71	19.50	11.70	3.19
15-19	22.58	3.23	11.29	7.89	6.14	0.88
20-24	16.00	8.00	4.00	5.26	2.63	0.00
25-30	0.00	0.00	0.00	5.26	0.00	0.00
30+	10.00	0.00	0.00	0.00	0.00	0.00

Therefore, it is possible that the potholes up close are large and do not have as many multiple detections as those that are further away from the camera. The best results was found, in general, to be in the 5-9 m range. The resolution and size of these potholes are possibly the easiest for the classifiers to detect. Lastly, as expected, the percentage of potholes that could be detected reduces as the distance from the camera increases. At roughly 25+ m, it can be seen that potholes can no longer be detected by most of the algorithms. The overall results are not particularly good, but dependent on how one evaluates the system, it can be speculated that potholes cannot really be reliably detected past 15 m by the LBP cascade classifier and SVM+HOG classifier. In both the simple scenario and complex scenario, it can be seen that by using this metric, the image processing algorithm performs the best of the classifiers in each scenario, as it detects more potholes. It does, however, also detect more false positives (Figure 5.7.1 and Figure 5.7.2).

## 5.9 CLASSIFIER TIMING PERFORMANCE

One of the aims of developing a pothole detection system was determining the viability of a real-time solution. In an effort to investigate this, each of the optimum classifiers (based on F1 score) was timed while detecting potholes in their respective test sets. These time trials were performed on an Intel Core i5 3.1 GHz computer with 8 Gig RAM installed.

The average time per classifier per photo is indicated in Table 5.9.1. It must be noted that these are the F1 score optimum settings for each classifier and that, for example, when setting the `hit_threshold` to 0.0 for the SVM+HOG (complex) it took significantly longer to process each photo. From the table it can be seen that the SVM+HOG in both the simple and complex scenario requires roughly 5 times longer to process an image than any of the other classifiers regardless of the scenario. The parameters for the HOG descriptor used was given in Table 4.4.1, and as these values are small in comparison to the resolution of the image, it requires more time to calculate the HOG features for each image than the LBP features. Also, due to the simple nature of the LBP feature calculations, these features are fast to calculate [115]. The timing performance of the LBP cascade and the image processing algorithms was found to be approximately the same. Should a real-time solution be required, the SVM+HOG classifier in its current form will not be able meet these standards and it would be better to implement either the LBP cascade classifier or image processing algorithms.

One of the factors that determine the amount of time it takes to process the image is the number of detections that the classifier outputs. More detections, whether true positives or false positives, result in a longer processing time than if the number of detections was lower. These detections are measured by the precision of the classifiers. For all of the complex scenario classifiers, the precision was generally much lower than their corresponding classifiers in the simple scenario. Therefore, many more false negatives were detected in the complex scenario, which would imply that it should take longer to process each image. From Table 5.9.1, however, it can be seen that this is not really the case. An explanation for this can be determined by examining Figure 5.8.1 and Figure 5.8.2. From these figures, it can clearly be seen that the number of detected potholes for the simple scenario was higher than that in the complex scenario for a given classifier. This leads to less detections in the complex scenario and therefore decreases the time required to process each photo.

**CHAPTER 5 – RESULTS****Table 5.9.1** Timing performance of the algorithms. The table indicates the average time it took for each classifier in its respective scenario (simple/complex) to detect potholes in a single image.

<b>Classifier</b>	<b>Average processing time per photo (s)</b>
LBP cascade (simple)	0.3569
LBP cascade (complex)	0.4934
SVM+HOG (simple)	1.8112
SVM+HOG (complex)	1.6735
Image processing (simple)	0.3646
Image processing (complex)	0.3709

When a driver perceives an unexpected but common occurrence such as a pothole in the road, it takes roughly 1.25 s to react [116]. This time must be added to the estimated processing time per image for the chosen algorithm. A scenario to determine the maximum speed at which a vehicle could travel and still detect potholes could be as follows: 1.25 s reaction time, LBP cascade (simple scenario), and a pothole detection threshold of roughly 10% (from Table 5.8.1) which results in a distance of 15 m. The total time for this scenario is equal to 1.6069 s. Therefore, assuming a constant vehicle speed, the maximum speed at which the vehicle can travel and yield the required pothole detection results is roughly 34 km/h. This speed is relatively slow as it is equal to half of that allowed in an urban area in South Africa. Therefore, pothole detection for real-time applications can only be deployed at slow speeds for human drivers. For autonomous vehicles the reaction time should be significantly faster than those of humans [117]. Therefore, autonomous vehicles should be able to detect potholes at higher speeds.

**5.10 CONCLUSION**

The results of the investigation was presented in this chapter. Various performance metrics were discussed in Section 5.2 and it was determined that precision/recall curves would be used to evaluate the classifiers' performance. It was also determined that in order to measure the most

## CHAPTER 5 – RESULTS

---

suitable setting for each classifier, the best precision/recall pair must be selected. The selected method to achieve this, is the F1 score. In Section 5.3, the necessary terminology and program settings to perform the tests were discussed. Section 5.4 discusses the results of the developed image processing algorithm for both the simple and complex scenario. The simple and complex scenario results for the LBP cascade classifier was given in Section 5.5. The last classifier that was investigated is the SVM+HOG classifier and the results of this classifier is presented in Section 5.6. A comparison based on the F1 score per scenario for all three classifiers is given in Section 5.7 and determined that the best classifier in the simple scenario was the SVM+HOG classifier. For the complex scenario, however, it was found that the LBP cascade classifier gave the best results.

The precision/recall point that had the highest F1 score for each classifier per scenario was evaluated in terms of the distance at which the classifier could detect potholes. It was found that the highest number of potholes could be detected in the range of 5-9.99 m.

Lastly, the timing performance of the classifiers were measured for the highest F1 score precision/recall point. From this it was found that the SVM+HOG classifier was much slower than the LBP cascade classifier and the image processing algorithm. The image processing algorithm and LBP cascade classifier' timing performance is roughly equal and therefore, either of these is recommended for a real-time application.



# CHAPTER 6

## CONCLUSION

### 6.1 SUMMARY OF WORK

This thesis investigates the feasibility of detecting potholes by using images. Since no suitable pothole footage could be found, a pothole database was created by driving a vehicle with a camera mounted to the inside of the windscreen around. The camera then captured images of the road. These images were annotated manually offline and analysed by the developed image processing and machine learning algorithms (LBP cascade classifier and SVM+HOG classifier).

It was also decided that due to the nature of the pothole problem, a simple scenario and complex scenario would be investigated. The simple scenario represents the ideal case of pothole detection where only open clear roads (with no/little shadows) with potholes are considered. Therefore, in this scenario, it is possible to extract only the road area in an image and detect potholes in this region. The complex scenario represents the real-world conditions that will be experienced when driving in dry weather. This scenario has a mix of images where some have mixed lighting conditions and others are clear. In this scenario, the road could not be extracted sufficiently. Therefore, the images were cropped to a fixed region where the road was expected to be. Separate training and test sets for both scenarios were created.

Lastly, to identify the effect of the pothole distance on the various systems' performance, three depth estimation techniques with emphasis on monocular vision were considered. An experiment was conducted to determine the most accurate among these depth estimation techniques, leading to the use of the cross ratio in this thesis.

### 6.2 CONCLUSIONS

This section summarizes the findings of the thesis and is divided into three sub-sections namely: pothole detection, depth estimation techniques and pothole detection range limitations.

---

## CHAPTER 6 – CONCLUSION

---

### 6.2.1 POTHOLE DETECTION

Pothole detection was implemented using three different techniques, namely image processing algorithms, LBP cascade classifier and SVM+HOG classifier.

The image processing algorithms developed in Section 4.2 – 4.3 were applied in both the simple and complex scenarios described earlier and the results are presented in Section 5.4. Although the image processing algorithms were developed to be used only in conjunction with open road conditions where the road could be successfully extracted (simple scenario), the image processing algorithms were also tested against the complex scenario where the lighting conditions varied so vastly that the road could not always be extracted successfully. This was done to provide thorough comparative results.

The theory of the LBP cascade classifier was discussed in Section 2.3.1 and Section 2.4.2. The selection of the size of the detector window was discussed in Section 4.4. The results of this classifier for both the simple and complex scenarios is presented in Section 5.5.

The last classifier that was investigated was the SVM+ HOG classifier. An overview of this classifier's theory was discussed in Section 2.3.2 and Section 2.4.3. The selection of the size of the parameters in the HOG descriptor was discussed in Section 4.4. This classifier was applied to the two different scenarios and the results is given in Section 5.6.

When the behaviour of all of the classifiers are observed at their optimal settings (F1-score point), it was found that, to a certain extent, potholes can be detected in images and can be observed in Table 5.7.1 and Table 5.8.1. Therefore, the feasibility of pothole detection has been investigated and **Research objective 1** has been satisfied.

By using precision/recall curves and the F1 score performance metric, the results of each classifier could be determined. Using the highest F1 score of each classifier for its scenario, allowed for the comparison of the classifiers.

- For the simple scenario from highest scoring to lowest scoring:
  - SVM+HOG (F1 score of 0.372942)
  - Image processing (F1 score of 0.210729)
  - LBP cascade (F1 score of 0.169969)

## CHAPTER 6 – CONCLUSION

---

From these results it can be seen that the SVM+HOG classifier performed the best in ideal pothole detection conditions. Therefore, comparing the different approaches and features for an ideal scenario, it can be seen that the HOG features are therefore most suited to detecting an object with a high variety in shape in an environment where there are not many other similar objects.

- For the complex scenario from highest scoring to lowest scoring:
  - LBP cascade (F1 score of 0.109457)
  - SVM+HOG (F1 score of 0.091088)
  - Image processing (F1 score of 0.060033)

From these results it can be seen that the LBP cascade classifier had the best performance in the real-world conditions scenario. This is attributed to the LBP features' ability to extract features when the intensity of an image varies (see Figure 2.3.2).

As demonstrated in the paragraphs above, the precision/recall curves and F1 score metric allowed for the comparison of the classifiers. Therefore, **Research objective 2** is satisfied.

By inspecting Section 5.7, it is found that although the algorithms can detect potholes, these results are not comparable to the more successful results obtained when other objects such as people are detected.

### 6.2.2 DEPTH ESTIMATION TECHNIQUES

Three different types of depth estimation in monocular images were investigated in Section 3.4, namely the pinhole model, cross ratio and similarity triangles. From an experimental setup it was found that the cross ratio was the most accurate method for estimating the depth in monocular images for the setup required in this thesis. This setup assumes that the depth estimation is necessary in a plane (representing the road) that is parallel to the optical axis of the camera. It was also found that for the depth estimation results to be more accurate towards the edges of the image, it is crucial to remove the fisheye effect in the image.

---

## CHAPTER 6 – CONCLUSION

---

### 6.2.3 POTHOLE DETECTION RANGE LIMITATIONS

The range at which potholes can be detected is important as it will determine how realisable a full system is, especially in terms of a driverless vehicle. The maximum range at which potholes can be detected is a major factor in determining at what maximum speed the vehicle detecting the potholes can travel at while still detecting potholes. The range analysis of each of the classifiers is discussed in Section 5.8. Another factor that will have an impact on the maximum speed that the vehicle can travel at is the processing time it takes for a particular classifier to process and detect potholes within an image. The faster the image rate of the algorithm, the faster the vehicle can travel and vice versa. The timing information regarding each of the classifiers for each scenario is given in Section 5.9.

The first conclusions that was evident from the range analysis was that almost all of the classifiers' performance peaked at a range of 5-9 m. When potholes were closer than 5 m, the algorithms are suspected to fail because of different reasons. In the case of the machine learning algorithms (LBP cascade classifier and SVM with HOG features classifier), it is possible that potholes closer than 5 m are much larger than the detector window and therefore do not have enough multiple detections to be deemed true positives. Considering the internal operation of the image processing algorithms, it was found that the potholes that were too close tended to be absorbed into the outer contour due to the dilation function and would therefore not be detected by the algorithm.

The second conclusion that could be made from the range analysis was that after the performance peak at a range of 5-9 m, it was clear that the performance dropped per range bracket until the range 25-29 m. After 30+ m, it was not possible for almost all of the algorithms to detect potholes any further.

Therefore, the effects of the distance at which potholes occur from the camera on the classifiers have been investigated, which satisfies **Research objective 3**.

### **6.3 FUTURE WORK**

It has been found that pothole detection is a very complicated matter which implies that more complex features and techniques need to be used in future to solve this problem more effectively. It is suspected that a neural network or deep learning algorithm will improve the performance of the pothole detection system. However, in order to investigate such sophisticated algorithms, it is first recommended to increase the number of positive samples. For example, in the instance of handwritten digit recognition, the MNIST database contains 60 000 training examples and 10 000 test samples. The pothole databases that was created in this thesis, contained roughly 5 000 samples in total. When it is attempted to detect an object with a high variety in shape, colour and general appearance, a very large number of positive samples will certainly help.

To reduce the occurrence of false positives, it is recommended to develop an improved algorithm that would extract the road surface. The main focus of the newly developed algorithm should be to extract the road surface accurately, especially in instances where a section of the road is occluded by tree shadows.

From the range analysis, it can be seen that all of the algorithms do not function very well at distances past 10 m (34.8% at best). Therefore, taking the maximum speed of the vehicle in account and for pothole detection to become more viable for real-time detection in driverless vehicles, this range needs to be increased substantially. It is possible that a higher resolution camera could solve this problem, although this will significantly increase the processing power required to detect potholes in the system. Also, if it is practically possible, mounting the camera another metre higher could provide the camera with a farther outlook and could possibly increase the range at which the potholes could be detected. Another option would be to investigate the effects of affine projections in the road plane. By converting the affine road section into a top-view scenario, it might also be possible to detect the potholes better as their shape might not differ as much from a top view. Currently, the same pothole's shape varies as its distance from the camera is varied. In theory, with the affine transformation, it would not matter at what distance the pothole occurs from the camera as the same pothole should always have the same shape throughout.

# APPENDIX A – CASCADE CLASSIFIER SETTINGS

The procedure for creating the positives samples for a cascade classifier, as well as training a cascade classifier in OpenCV is explained in this appendix.

The first utility used is `opencv_createsamples` and a table of its inputs is given in Table A.1. This utility is used to prepare the positive samples that will be used in the training phase of the cascade classifier. The command line that was used for `opencv_createsamples` is:

```
C:\opencv\build\x64\vc11\bin\opencv_createsamples.exe -info positive.txt -vec posVec.vec
-num 3074 -w 32 -h 16 PAUSE
```

Note that the samples were not rotated, as it was thought that this would negatively impact the classifier.

The output vec-file created by `opencv_createsamples` is input to `opencv_traincascade`. A table explaining the inputs of `opencv_traincascade` is given in Table A.2. This utility will output the .xml file created that contains the structure of the cascade classifier. This output cascade is input to the `detectMultiScale` function discussed in Appendix B. The command used in `opencv_traincascade` is:

```
C:\opencv\build\x64\vc11\bin\opencv_traincascade.exe -data classifierLBP -vec posVec.vec
-bg negative.txt -numStages 21 -miniHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 3074
-numNeg 6192 -w 32 -h 16 -featureType LBP -precalcValBufSize 1000 -precalcIdxBufSize 1000
-maxWeakCount 500
```

The `miniHitRate`, `maxFalseAlarmRate` and `maxWeakCount` parameters were kept to the standard values given in [118].

**APPENDIX A – CASCADE CLASSIFIER****Table A.1 Parameters of `opencv_createsamples`. Input parameters as given in [118].**

Input	Description
<code>-vec &lt;vec_file_name&gt;</code>	Name of the output file containing the positive samples for training.
<code>-img &lt;image_file_name&gt;</code>	Source object image.
<code>-bg &lt;background_file_name&gt;</code>	Background description file; contains a list of images which are used as a background for randomly distorted versions of the object.
<code>-num &lt;number_of_samples&gt;</code>	Number of positive samples to generate.
<code>-bgcolor &lt;background_color&gt;</code>	Background color (currently grayscale images are assumed); the background color denotes the transparent color. Since there might be compression artifacts, the amount of color tolerance can be specified by <code>-bgthresh</code> . All pixels withing <code>bgcolor-bgthresh</code> and <code>bgcolor+bgthresh</code> range are interpreted as transparent.
<code>-bgthresh</code> <code>&lt;background_color_threshold&gt;</code>	
<code>-inv</code>	If specified, colors will be inverted.
<code>-randinv</code>	If specified, colors will be inverted randomly.
<code>-maxidev &lt;max_intensity_deviation&gt;</code>	Maximal intensity deviation of pixels in foreground samples.
<code>-maxxangle &lt;max_x_rotation_angle&gt;</code> <code>-maxyangle &lt;max_y_rotation_angle&gt;</code> <code>-maxzangle &lt;max_z_rotation_angle&gt;</code>	Maximum rotation angles must be given in radians.
<code>-show</code>	Useful debugging option. If specified, each sample will be shown. Pressing <code>Esc</code> will continue the samples creation process without.
<code>-w &lt;sample_width&gt;</code>	Width (in pixels) of the output samples.
<code>-h &lt;sample_height&gt;</code>	Height (in pixels) of the output samples.
<code>-pngoutput</code>	With this option switched on <code>opencv_createsamples</code> tool generates a collection of PNG samples and a number of associated annotation files, instead of a single <code>vec</code> file.

**APPENDIX A – CASCADE CLASSIFIER****Table A.2 Parameters of `open_traincascade`. Input parameters as given in [118].**

Input	Parameter
<code>-data &lt;cascade_dir_name&gt;</code>	Where the trained classifier should be stored.
<code>-vec &lt;vec_file_name&gt;</code>	vec-file with positive samples (created by <code>opencv_createsamples</code> utility).
<code>-bg &lt;background_file_name&gt;</code>	Background description file.
<code>-numPos &lt;number_of_positive_samples&gt;</code> <code>-numNeg &lt;number_of_negative_samples&gt;</code>	Number of positive/negative samples used in training for every classifier stage.
<code>-numStages &lt;number_of_stages&gt;</code>	Number of cascade stages to be trained.
<code>-precalcValBufSize</code> <code>&lt;precalculated_vals_buffer_size_in_Mb&gt;</code>	Size of buffer for precalculated feature values (in Mb).
<code>-precalcIdxBufSize</code> <code>&lt;precalculated_idxs_buffer_size_in_Mb&gt;</code>	Size of buffer for precalculated feature indices (in Mb). The more memory you have the faster the training process.
<code>-baseFormatSave</code>	This argument is actual in case of Haar-like features. If it is specified, the cascade will be saved in the old format.
<code>-acceptanceRatioBreakValue</code>	This argument is used to determine how precise your model should keep learning and when to stop. A good guideline is to train not further than $10e-5$ , to ensure the model does not overtrain on your training data. By default this value is set to -1 to disable this feature.
<code>-stageType &lt;BOOST(default)&gt;</code>	Type of stages. Only boosted classifier are supported as a stage type at the moment.
<code>-featureType&lt;{HAAR(default), LBP}&gt;</code>	Type of features: HAAR - Haar-like features, LBP - local binary patterns.
<code>-w &lt;sampleWidth&gt;</code> <code>-h &lt;sampleHeight&gt;</code>	Size of training samples (in pixels). Must have exactly the same values as used during training samples creation ( <code>opencv_createsamples</code> utility).
<code>-bt &lt;{DAB, RAB, LB, GAB(default)}&gt;</code>	Type of boosted classifiers: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost.
<code>-minHitRate &lt;min_hit_rate&gt;</code>	Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as $(\text{min\_hit\_rate}^{\text{number\_of\_stages}})$ .
<code>-maxFalseAlarmRate</code> <code>&lt;max_false_alarm_rate&gt;</code>	Maximal desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as $(\text{max\_false\_alarm\_rate}^{\text{number\_of\_stages}})$ .
<code>-weightTrimRate &lt;weight_trim_rate&gt;</code>	Specifies whether trimming should be used and its weight. A decent choice is 0.95.
<code>-maxDepth &lt;max_depth_of_weak_tree&gt;</code>	Maximal depth of a weak tree. A decent choice is 1, that is case of stumps.
<code>-maxWeakCount &lt;max_weak_tree_count&gt;</code>	Maximal count of weak trees for every cascade stage. The boosted classifier (stage) will have so many weak trees ( $\leq \text{maxWeakCount}$ ), as needed to achieve the given <code>-maxFalseAlarmRate</code> .
<code>-mode &lt;BASIC (default)   CORE   ALL&gt;</code>	Selects the type of Haar features set used in training. BASIC use only upright features, while ALL uses the full set of upright and 45 degree rotated feature set.



# APPENDIX B – DETECTOR SETTINGS

In this appendix the settings for the `detectMultiScale` function in OpenCV is given. There are two different implementations of the function which depends on which type of classifier is being used.

The first type of classifier that this function was used with is the cascade classifier. This classifier's implementation of `detectMultiScale` is given in Table B.1. For this function, a trained classifier file and input image is required. The `scaleFactor` and `minNeighbors` setting must be set, and the `minSize` and `maxSize` can be omitted (as was the case). The output (objects) is a vector that contains the rectangles where the suspected detection regions are.

**Table B.1 Cascade classifier detection settings [112]**

Input	Description
<code>cascade</code>	Cascade classifier (OpenCV 1.x API only). It can be loaded from XML or YAML file using <code>Load()</code> .
<code>image</code>	Matrix of the type <code>CV_8U</code> containing an image where objects are detected.
<code>objects</code>	Vector of rectangles where each rectangle contains the detected object.
<code>scaleFactor</code>	Parameter specifying how much the image size is reduced at each image scale.
<code>minNeighbors</code>	Parameter specifying how many neighbours each candidate rectangle should have to retain it.
<code>minSize</code>	Minimum possible object size. Objects smaller than that are ignored.
<code>maxSize</code>	Maximum possible object size. Objects larger than that are ignored

The second implementation of the `detectMultiScale` function applies to the HOG descriptor and is given in Table B.2. This function is called by the command `hog.detectMultiScale(...)` where `hog` is the created input HOG descriptor by using the parameters given in Table 4.4.1. It requires an input image and setting the `hit_threshold`, `win_stride`, `padding`, `scale0` and `group_threshold`. It outputs a vector of rectangles with the detected regions in the `found_locations` parameter.

**APPENDIX B – DETECTOR SETTINGS**

---

**Table B.2 HOG detection settings [114].**

<b>Input</b>	<b>Description</b>
img	Source image
found_locations	Detected objects boundaries
hit_threshold	Threshold for the distance between features and SVM classifying plane.
win_stride	Window stride. It must be a multiple of block stride.
padding	Mock parameter to keep the CPU interface compatibility. It must be (0,0).
scale0	Coefficient of the detection window increase.
group_threshold	Coefficient to regulate the similarity threshold. When detected, some objects can be covered by many rectangles. 0 means not to perform grouping.

# APPENDIX C – PRACTICAL DEPTH ESTIMATION CONSIDERATIONS

## C.1 FISHEYE REMOVAL

When depth estimation is required within images, it is necessary to remove distortion in the images to ensure the integrity of the depth estimation results. The GoPro camera introduces a significant amount of fisheye distortion as demonstrated in Figure 3.7.3. The detailed methodology used for correcting the fisheye is discussed here.

The software that was used to correct the fisheye distortion is the Camera Calibration Toolbox for Matlab [87]. As the calibration procedure described in Section 3.7.3 was performed on images that had a resolution of 3680 x 2760 pixels, the fisheye distortion can only be applied to images with the same resolution. However, to perform fisheye correction at these high resolutions requires a significant amount of time and processing and would not be feasible for a large test set. Therefore, it was decided that a more simplistic solution is needed and it was decided to perform the fisheye correction on a *pixel only* basis and not on the entire image.

To perform fisheye correction on a pixel at a time, LUTs from the original toolbox were created. This entailed creating a matrix of size 3680 x 2760 and each of the cells in the matrix were valued between 1 – 10156800. The fisheye correction was applied to this matrix and a new matrix with the updated pixel positions (and consequently values) was created. LUTs of the old and new indices were created by searching the values of the new matrix for the values in the old matrix.

The only pixels of interest are those at which the potholes occur and therefore the procedure to perform the fisheye correction per pothole is as follows:

1. Calculate the pothole's pixel reference point. This point is calculated by taking the y-value of the top left corner of the bounding box and adding the height of the pothole, as well as the y-value offset of the road cropping function (see Figure 3.4.1). To adjust the x-value of the pixel, the x-value of the top left corner of the bounding box is added to half of the width of the pothole. This new (x,y) pixel's values are doubled to compensate for the downscaling operation performed on the images.

## *APPENDIX C – PRACTICAL DEPTH ESTIMATION CONSIDERATIONS*

---

2. The calculated pothole's reference pixel is then used in a function that compares the old index to the new index and outputs the new location of the pixel.

Once this procedure is completed, the depth estimation can be performed.

### **C.2 PRACTICAL DEPTH ESTIMATION**

The depth estimation is performed by using the vanishing point in the image. A vanishing point is point where the parallel lines in an image converge and to an observer seems like a point at infinity [90]. The vanishing point is calculated by identifying two different lines in an image and selecting two different pixel points on each line. The two pixels per line are converted to homogeneous points and the cross product between them are calculated to determine the homogenous representation of their lines. The cross product of the two homogenous lines are then calculated which yields the vanishing point. The reference vanishing point was calculated for the calibration images used in Section 3.7. This is because when the optical axis of the camera is aligned exactly to the horizontal surface of the ground, the vanishing point of the lines on the surface, will be at the centre of the image [119]. Then, by applying the cross-ratio formula, a LUT could be created that equated a particular y-value of a pixel to a particular estimated depth. Note that for the vanishing point calculations to be valid, all forms of distortion, including fisheye distortion needs to be removed from the entire image.

A matter of practical concern, however, was how to ensure the integrity of the depth estimation, considering that the camera was removed when the battery died/memory card was full during the period when the footage was collected. Although great care was taken to replace the camera at the exact same location and angle facing the road, this was not always the case and small variations in the tilt occurred. Therefore, an additional procedure was necessary to correct for this matter.

As the camera was not moved in-between the times when it had to be removed, the procedure required that the images be grouped together per camera position/time frame. After this step, a random number of images were collected per grouped selection as a subset. For each subset, fisheye correction was implemented on each of the images. Thereafter, the vanishing point of each of the images were manually obtained.

---

*APPENDIX C – PRACTICAL DEPTH ESTIMATION CONSIDERATIONS*

---

The average vanishing point per camera position was calculated per subset and a LUT was created which contained all of the names of the images in the datasets (simple and complex separately) with the corresponding average vanishing point of each image. This average vanishing point was compared to the reference vanishing point obtained in the ideal case and an offset  $y$ -value could be determined to adjust for the depths accordingly.

Considering how small the variations were when the camera was moved, and the fact that as the vehicle drove around, small variations in the plane of the road in the images would occur due to the road surface not being exactly level, this procedure was the best thought practical solution without the use of laser range finder. Therefore, by using the average vanishing point per camera position, these effects can be reasonably minimised for a real-world solution.

# BIBLIOGRAPHY

- [1] H. Wittmann, “Total Costs of Logistics in South Africa Need to be Reduced,” 2010.
- [2] C. B. Environment, December 2010. [Online]. Available:  
[http://www.csir.co.za/pothole\\_guides/docs/Pothole\\_CSIR\\_tech\\_guide.pdf](http://www.csir.co.za/pothole_guides/docs/Pothole_CSIR_tech_guide.pdf).
- [3] [Online]. Available: <http://www.roadscanners.com/index.php/roadscanners>.
- [4] J. Deaton and K. Hall-Geisler, “How Driverless Cars Will Work,” HowStuffWorks Auto, [Online]. Available: <http://auto.howstuffworks.com/under-the-hood/trends-innovations/driverless-car.htm>. [Accessed 19 November 2015].
- [5] “Google Self-Driving Car Project,” Google, [Online]. Available:  
<https://www.google.com/selfdrivingcar/>. [Accessed 19 November 2015].
- [6] G. Bradski, “OpenCV library,” *Dr. Dobbs's Journal of Software Tools*, 2000.
- [7] “What is colour space,” ArcSoft, [Online]. Available:  
<http://www.arcsoft.com/topics/photostudio-darkroom/what-is-color-space.html>.  
[Accessed 22 November 2015].
- [8] T. Harris and W. Fenlon, “How Light Emitting Diodes Work,” HowStuffWorks, [Online]. Available: <http://electronics.howstuffworks.com/led5.htm>. [Accessed 6 September 2015].
- [9] “Term: colour channel,” Federal Agencies Digitization Guidelines Initiative, [Online]. Available: <http://www.digitizationguidelines.gov/term.php?term=colorchannel>.  
[Accessed 22 November 2015].
- [10] S. Sangwine and R. Horne, “Chapter 4: Representations of colour images in different colour spaces,” in *The Colour Image Processing Handbook (Optoelectronics, Imaging and Sensing)*, New Jersey, Springer-Verlag New York, Inc, 1998, pp. 67-69.

- [11] S. Sural, G. Qian and S. Pramanik, "Segmentation and Histogram Generation Using the HSV Colour Space for Image Retrieval," in *International Conference on Image Processing Volume 2*, Rochester, New York, 2002.
- [12] K. v. d. Sande, T. Gevers and C. Snoek, "Evaluating Color Descriptors for Object and Scene Recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1582-1596, 2010.
- [13] "ROHAN Academic Computing," San Diego State University, 29 July 2011. [Online]. Available: <http://www-rohan.sdsu.edu/doc/matlab/toolbox/images/color11.html>. [Accessed 5 September 2015].
- [14] G. Bradski and A. Kaehler, in *Learning OpenCV*, California, O'Reilly Media Inc., 2008.
- [15] K. Hunt, *The Art of Image Processing with Java*, Florida: CRC Press, 2010.
- [16] L. Shapiro and G. Stockman, *Computer Vision*, New Jersey: Prentice Hall, 2001.
- [17] P. Fuller, "Patrick Fuller's blog," 25 November 2012. [Online]. Available: <http://patrick-fuller.com/gaussian-blur-image-processing-for-scientists-and-engineers-part-4/>. [Accessed 1 September 2015].
- [18] "Morphology fundamentals: Dilation and Erosion," Mathworks, 2015. [Online]. Available: <http://www.mathworks.com/help/images/morphology-fundamentals-dilation-and-erosion.html>. [Accessed 15 November 2015].
- [19] M. Goyal, "Morphological Image Processing," *International Journal of Computer Science and Technology (IJCST)*, vol. 2, no. 4, pp. 161-165, 2011.
- [20] J. Parker, *Algorithms for Image Processing and Computer Vision 2nd Edition*, Indianapolis: Wiley Publishing, Inc., 2011.
- [21] J. Prewitt, "Object Enhancement and Extraction," in *Picture Processing and Psychopictorics*, B. Lipkin and A. Rosenfeld, eds., 1970, 1970.
- [22] L. Roberts, "Machine Perception of Three-Dimensional Solids," in *Optical and Electro-Optical Information Processing*, J.T. Tippett, et al., MIT Press, 1965.

- [23] I. Sobel and G. Feldman, "A 3 x 3 Isotropic Gradient Operator for Image Processing," in *Pattern Classification and Scene Analysis*, R. Duda and P. Hart, eds., Wiley, 1973, pp. 271-272.
- [24] H. Scharr, "Optimal Operators In Digital Image Processing," in *Ph.D. thesis, Interdisciplinary Center for Scientific Computing, Ruprecht-Karls-Universität, Heidelberg*, <http://www.fz>, 2000.
- [25] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1986.
- [26] P. Kalra, "CSL783: Digital Image Processing, Department of Computer Science and Engineering, Indian Institute of Technology," September 2015. [Online]. Available: <http://www.cse.iitd.ernet.in/~pkalra/csl783/canny.pdf>. [Accessed 3 September 2015].
- [27] "OpenCV 3.0.0-dev," 2 September 2015. [Online]. Available: [http://docs.opencv.org/master/da/d5c/tutorial\\_canny\\_detector.html#gsc.tab=0](http://docs.opencv.org/master/da/d5c/tutorial_canny_detector.html#gsc.tab=0). [Accessed 3 September 2015].
- [28] A. Aichert, "Technische Universität München," [Online]. Available: <http://homo.in.tum.de/~aichert/featurepres.pdf>. [Accessed 20 August 2015].
- [29] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
- [30] H. Bay, T. Tuytelaars and L. v. Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359, 2008.
- [31] A. Rosebrock, "PyImageSearch," 16 July 2015. [Online]. Available: <http://www.pyimagesearch.com/2015/07/16/where-did-sift-and-surf-go-in-opencv-3/>. [Accessed 25 August 2015].
- [32] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 511-518, 2001.



- [33] T. Ojala, M. Pietikainen and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51-59, 1996.
- [34] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.
- [35] N. Dalal, "Finding People in Images and Videos, Phd Thesis," Institut National Polytechnique de Grenoble, Rhône-Alpes, 2006.
- [36] "Is HaarDetectObjects rotation invariant?," OpenCV, 29 August 2012. [Online]. Available: <http://answers.opencv.org/question/1889/is-haardetectobjects-rotation-invariant/>. [Accessed 20 November 2015].
- [37] Z. Guo, L. Zhang and D. Zhang, "Rotation invariant texture classification using LBP variance (LBPV) with," *Pattern Recognition*, vol. 43, no. 3, pp. 706-719, 2010.
- [38] M. Pietikäinen, A. Hadid, G. Zhao and T. Ahonen, *Computer Vision Using Local Binary Patterns*, Springer, 2011.
- [39] M. Barani, K. Faez and F. Jalili, "Implementation of Gabor Filters Combined with Binary Features for Gender Recognition," *International Journal of Electrical and Computing Engineering (IJECE)*, vol. 4, no. 1, pp. 108-115, 2014.
- [40] "Understanding gamma correction," Cambridge in colour, 2015. [Online]. Available: <http://www.cambridgeincolour.com/tutorials/gamma-correction.htm>. [Accessed 22 November 2015].
- [41] G. Finlayson and R. Xu, "Gamma Comprehensive Normalisation," in *10th Color Imaging Conference (CIC): Color Science and Engineering Systems, Technologies, Applications*, Scottsdale, Arizona, USA, 2002.
- [42] J. Brauer, "HOG descriptor computation and visualization," Job-related website of Jürgen Brauer, 19 January 2015. [Online]. Available: [http://www.juergenwiki.de/work/wiki/doku.php?id=public:hog\\_descriptor\\_computation\\_and\\_visualization](http://www.juergenwiki.de/work/wiki/doku.php?id=public:hog_descriptor_computation_and_visualization). [Accessed 6 September 2015].

- [43] S. Ullman, "Chapter 1: Object Recognition," in *High-level vision, Object Recognition and Visual Cognition*, Cambridge, Massachusetts, The MIT Press, 2000, pp. 1-12.
- [44] V. Vapnik, Estimation of Dependences Based on Empirical Data [in Russian], Nauka, Moscow: (English translation: Springer Verslag, New York, 1982), 1779.
- [45] Y. LeCun, L. Bottou, G. Orr and K. Muller, "Efficient backprop," in *Neural Networks - Tricks of the Trade*, Berlin Heidelberg, Springer-Verlag, 1998, pp. 5-50.
- [46] M. Kearns, "Thoughts on hypothesis boosting," Unpublished manuscript, 1988.
- [47] Y. Freund and R. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, 1999.
- [48] R. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197-227, 1990.
- [49] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.
- [50] V. Vapnik and O. Chapelle, "Bound on error expectation for support vector machines," *Neural Computation*, vol. 12, no. 9, pp. 2013-2036, 2000.
- [51] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121-167, 1998.
- [52] C. Manning, P. Raghavan and H. Schütze, "Chapter 15: Support vector machines and machine learning on documents," in *An Introduction to Information Retrieval*, Cambridge, England, Cambridge UP, 2009, pp. 319-348.
- [53] O. Chapelle, V. Vapnik, O. Bousquet and S. Mukherjee, "Choosing Multiple Parameters for Support Vector Machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131-159, 2002.
- [54] R. Pillay, "Sourceforge," [Online]. Available: <http://iipimage.sourceforge.net/documentation/images/>. [Accessed 22 August 2015].

- 
- [55] S. Gould, "How to train a sliding-window object detector," 31 January 2009. [Online]. Available: <http://ai.stanford.edu/~sgould/svl/howto/svlTrainingObjectDetectors.pdf>. [Accessed 28 March 2015].
- [56] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden and H. Balakrishnan, "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring," in *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, Breckenridge, U.S.A., 2008.
- [57] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *International Conference on Distributed Computing in Sensor Systems and Workshops*, Barcelona, 2011.
- [58] P. Burrows, July 2013. [Online]. Available: <http://www.bloomberg.com/news/2013-07-21/high-end-smartphone-boom-ending-as-price-drop-hits-apple.html>.
- [59] J. Karuppuswamy, V. Selvaraj, M. M. Ganesh and E. L. Hall, "Detection and Avoidance of Simulated Potholes in Autonomous Vehicle Navigation in an Unstructured Environment," in *Proc. SPIE 4197, Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision*, 2000.
- [60] L. Huidrom, L. Das and S. Sud, "Method for Automated Assessment of Potholes, Cracks and Patches from Road Surface Video Clips," in *2nd Conference of Transportation Research Group of India*, Tah Mahal, 2013.
- [61] H. Lokeshor, L. Das and S. Goel, "Robust Mthod for Automated Segmentation of Frames with/without Distress from Road Surface Video Clips," *Journal of Transportation Engineering*, vol. 140, no. 1, pp. 31-41, 2014.
- [62] C. Koch and I. Brilakis, "Improving Pothole Recognition through Vision Tracking for Automated Pavement Assessment," in *18th International EG-ICE Workshop*, Enschede, Netherlands, 2011.
- [63] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507-515, August 2011.

- [64] K. Wang, "Automated Pavement Distress Survey through Stereovision," Highway IDEA Program, Transportation Research Board of the National Academies, Arkansas, 2004.
- [65] M. Staniek, "Stereo vision techniques in the road pavement evaluation," in *XXVIII International Baltic Road Conference*, Vilnius, Lithuania, 2013.
- [66] February 2011. [Online]. Available: [http://openkinect.org/wiki/Hardware\\_info](http://openkinect.org/wiki/Hardware_info).
- [67] D. Joubert, A. Tyatyantsi, J. Mphahlehle and V. Manchidi, "Pothole Tagging System," in *4th Robotics and Mechatronics Conference of South Africa - RobMech*, Pretoria, 2011.
- [68] D. A. S. Lab, June 2013. [Online]. Available: [http://dasl.mem.drexel.edu/wiki/index.php/KINECT\\_in\\_direct\\_sunlight](http://dasl.mem.drexel.edu/wiki/index.php/KINECT_in_direct_sunlight).
- [69] L. Terblanche, "Potholes in the Vaal," Facebook page, Created 4 May 2015. [Online]. Available: <https://www.facebook.com/potholesinthevaal/timeline>. [Accessed 20 September 2015].
- [70] J. Carucci, "GoPro Cameras: Time-Lapse Mode," For dummies, a Wiley brand, [Online]. Available: <http://www.dummies.com/how-to/content/gopro-cameras-timelapse-mode.html>. [Accessed 20 September 2015].
- [71] B. Durand, "GoPro Introduces the Hero 3+," Underwater Photography Guide, 1 October 2013. [Online]. Available: <http://www.uwphotographyguide.com/gopro-hero3-black>. [Accessed 20 September 2015].
- [72] L. Wang, T. Yang and Y. Tian, "Crop Disease Leaf Image Segmentation Method Based on Color Features," in *First IFIP TC 12 International Conference on Computer and Computing Technologies in Agriculture (CCTA 2007)*, Wuyishan, China, 2007.
- [73] C. M. Gautam, S. Sharma and J. S. Verma, "A GUI for Automatic Extraction of Signature from Image Document," *International Journal of Computer Applications*, vol. 54, no. 15, pp. 13-19, 2012.

- [74] K. Setarehdan and S. Singh, "Chapter 9. Digital Mammography Segmentation," in *Advanced Algorithmic Approaches to Medical Image Segmentation*, London, Springer-Verlag, 2002, p. 476.
- [75] B. Fernando, "OpenCV Haartraining," Shehan's blog, 19 June 2011. [Online]. Available: <http://www.tectute.com/2011/06/opencv-haartraining.html>. [Accessed 25 February 2015].
- [76] T. O. U. Guide, "OpenCV," 25 June 2014. [Online]. Available: [http://www.docs.opencv.org/opencv\\_user.pdf](http://www.docs.opencv.org/opencv_user.pdf). [Accessed 15 March 2015].
- [77] V. Singh, A. Rai, N. Hemanth, D. Rohit and A. Mukherjee, "Algorithms for real time detection and depth calculation of obstacles by autonomous robots," in *IEEE Conference on Robotics, Automation and Mechatronics*, Chengdu, 2008.
- [78] A. Criminisi, I. Reid and A. Zisserman, "Single view metrology," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123-148, 2000.
- [79] I. Rapp, "Motion capture actors: body movement tells the story," [Online]. Available: [http://www.nycastings.com/dmxreadyv2/blogmanager/v3\\_blogmanager.asp?post=motioncaptureactors](http://www.nycastings.com/dmxreadyv2/blogmanager/v3_blogmanager.asp?post=motioncaptureactors). [Accessed 20 May 2015].
- [80] A. Bhatti, Current advancements in stereo vision, InTech, 2012.
- [81] A. Joglekar, D. Joshi, R. Khemani, S. Nair and S. Sahare, "Depth estimation using monocular camera," *International journal of computer science and information technologies*, vol. 2, no. 4, pp. 1758-1763, 2011.
- [82] G. Stein, O. Mano and A. Shashua, "Vision-based ACC with a single camera: bounds on range and range rate accuracy," in *Intelligent vehicles symposium*, Columbus, 2003.
- [83] A. Saxena, S. Chung and A. Y. Ng, "3D depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 53-69, 2008.
- [84] B. Lui, S. Gould and D. Koller, "Single image depth estimation from predicted semantic labels," in *IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, 2010.

- [85] A. Rahimi, H. Moradi and R. A. Zoroofi, "Single image ground plane estimation," in *20th IEEE International Conference on Image Processing*, Melbourne, 2013.
- [86] R. Szeliski, *Computer Vision - Algorithms and Applications*, London: Springer, 2011.
- [87] J. Bouguet, "Camera calibration toolbox for Matlab," [Online]. Available: [http://www.vision.caltech.edu/bougetj/calib\\_doc/](http://www.vision.caltech.edu/bougetj/calib_doc/). [Accessed 21 April 2015].
- [88] R. Dragon and L. V. Gool, "Ground plane estimation using a hidden markov model," in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014.
- [89] J. Bird, *Basic engineering mathematics 4th Edition*, Elsevier, 2005.
- [90] A. Zisserman and R. Hartley, *Multiple view geometry in computer vision*, second edition, Cambridge, 2000.
- [91] Stanford Gablab, "Imaging Knowledge Base Wiki: Smoothing FAQ," MIT, 30 April 2008. [Online]. Available: <http://mindhive.mit.edu/book/export/html/112>. [Accessed 2 June 2014].
- [92] M. Raman and H. Aggarwal, "Study and Comparison of Various Image Edge Detection Techniques," *International Journal of Image Processing*, vol. 3, no. 1, pp. 1-11, 2009.
- [93] X. Cao, C. Wu, P. Yan and X. Li, "Linear SVM classification using boosting HOG features for vehicle detection in low-altitude airborne videos," in *18th IEEE International Conference on Image Processing*, Brussels, Belgium, 2011.
- [94] D. Llorca, R. Arroyo and M. Sotelo, "Vehicle logo recognition in traffic images using HOG features and SVM," in *16th IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, 2013.
- [95] J. Sochor, "Traffic Analysis From Video, Master's Thesis," Brno University Of Technology, 2014, 2014.
- [96] S. Mathe and C. Sminchisescu, "Actions in the Eye: Dynamic Gaze Datasets and Learnt Saliency Models for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1408-1424, 2014.

- [97] R. Gutierrez-Osuna, "Department of Computer Science and Engineering," A&M University, [Online]. Available: [research.cs.tamu.edu/prism/lectures/iss/iss\\_113.pdf](http://research.cs.tamu.edu/prism/lectures/iss/iss_113.pdf). [Accessed 14 October 2015].
- [98] J. Schneider, "Jeff Schneider, Tutorial 5," Carnegie Mellon University, 17 September 1998. [Online]. Available: <https://www.cs.cmu.edu/~schneide/tut5/node42.html>. [Accessed 14 October 2015].
- [99] "Support Vector Machines," OpenCV web site, 15 February 2015. [Online]. Available: [http://docs.opencv.org/modules/ml/doc/support\\_vector\\_machines.html](http://docs.opencv.org/modules/ml/doc/support_vector_machines.html). [Accessed 25 July 2015].
- [100] E. Alpaydin, "Chapter 19: Design and analysis of machine learning experiments," in *Introduction to Machine Learning, Second Edition*, Cambridge, Massachusetts, The MIT Press, 2010, pp. 475-515.
- [101] C. Hsu, C. Chang and C. Lin, "Department of Computer Science, National Taiwan University," 15 April 2010. [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>. [Accessed 20 August 2015].
- [102] "Cross validation is very slow in Grid search (libsvm)," Stackoverflow, [Online]. Available: <http://coderissues.com/questions/20719038/cross-validation-is-very-slow-in-grid-search-libsvm>. [Accessed 10 August 2015].
- [103] A. Kowalszyk, "SVM Tutorial," SVM Tutorial Blog, 19 October 2014. [Online]. Available: <http://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/>. [Accessed 12 October 2015].
- [104] S. Keerthi and C. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667-1689, 2003.
- [105] "Precision-Recall," Scikit learn, [Online]. Available: [http://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_precision\\_recall.html](http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html). [Accessed 25 September 2015].

- [106] “Differences between Receiver Operating Characteristic AUC (ROC AUC) and Precision Recall AUC (PR AUC),” Garbled Notes, 19 April 2014. [Online]. Available: <http://www.chioka.in/tag/machine-learning/page/2/>. [Accessed 25 September 2015].
- [107] S. Wang and X. Yao, “Multiclass Imbalance Problems: Analysis and Potential Solutions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 4, pp. 1119-1130, 2012.
- [108] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” in *ICML '06 Proceedings of the 23rd international conference on Machine learning*, Pittsburgh, Pennsylvania, 2006.
- [109] “Differences between Receiver Operating Characteristic AUC (ROC AUC) and Precision Recall AUC (PR AUC),” Garbled Notes, 19 4 2014. [Online]. Available: <http://www.chioka.in/differences-between-roc-auc-and-pr-auc/>. [Accessed 25 August 2015].
- [110] C. Goutte and E. Gaussier, “A Probabilistic Interpretation of Precision, Recall and F-score, with Implication for Evaluation,” [Online]. Available: [https://www.google.co.za/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0CDsQFjAEahUKEWjAp5n42ZjJAhUBCB0KHcaKAL0&url=http%3A%2F%2Fwww3.xrce.xerox.com%2Fcontent%2Fdownload%2F20797%2F148382%2Ffile%2Fxrce\\_eval.pdf&usg=AFQjCNHNqYb5gXjVSDukbpwcOkG](https://www.google.co.za/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0CDsQFjAEahUKEWjAp5n42ZjJAhUBCB0KHcaKAL0&url=http%3A%2F%2Fwww3.xrce.xerox.com%2Fcontent%2Fdownload%2F20797%2F148382%2Ffile%2Fxrce_eval.pdf&usg=AFQjCNHNqYb5gXjVSDukbpwcOkG). [Accessed 25 September 2015].
- [111] “PASCAL 2. Pattern Analysis, Statistical Modelling and Computational Learning,” [Online]. Available: <http://www.pascal-network.org/?q=node/15>. [Accessed 25 September 2015].
- [112] “Cascade Classification,” OpenCV, 1 June 2015. [Online]. Available: [http://docs.opencv.org/2.4/modules/objdetect/doc/cascade\\_classification.html](http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html). [Accessed 5 June 2015].
- [113] M. Dimashova, “Traincascade Error: Bad argument (Can not get new positive sample. The most possible reason is insufficient count of samples in given vec-file,” OpenCV Answers, 20 November 2012. [Online]. Available:



<http://answers.opencv.org/question/4368/traincascade-error-bad-argument-can-not-get-new-positive-sample-the-most-possible-reason-is-insufficient-count-of-samples-in-given-vec-file/#4474>. [Accessed 16 March 2015].

- [114] "Object Detection," OpenCV, 2015. [Online]. Available: [http://docs.opencv.org/2.4/modules/gpu/doc/object\\_detection.html](http://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html). [Accessed 30 July 2015].
- [115] V. Takala, T. Ahonen and M. Pietikäinen, "Block-Based Methods for Image Retrieval Using Local Binary Patterns," in *14th Scandinavian Conference, SCL4*, Joensuu, Finland, 2005.
- [116] M. Green, "'How Long Does It Take to Stop?' Methodological Analysis of Driver Perception-Brake Times," *Transportation Human Factors*, vol. 2, no. 3, pp. 195-216, 2000.
- [117] L. Bell, "Humans vs robots: Driverless cars are safer than human driven vehicles," *The Inquirer*, 23 September 2015. [Online]. Available: <http://www.theinquirer.net/inquirer/feature/2426988/humans-vs-robots-driverless-cars-are-safer-than-human-driven-vehicles>. [Accessed 28 November 2015].
- [118] "Cascade Classifier Training," OpenCV, 2015. [Online]. Available: [http://docs.opencv.org/2.4/doc/user\\_guide/ug\\_traincascade.html](http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html). [Accessed 25 November 2015].
- [119] P. Ward, "Chapter 4," in *Picture Composition for Film and Television, Second Edition*, Oxford, Elsevier Science, 2003, pp. 42-43.
- [120] S. Liao, X. Zhu, Z. Lei, L. Zhang and S. Li, "Learning Multi-scale Block Local Binary Patterns for Face Recognition," in *International Conference on Biometrics (ICB)*, 2007.
- [121] C. McCormick, "Gradient Vectors," *Computer Vision and Machine Learning Projects and Tutorials*, Wordpress, 7 May 2013. [Online]. Available: <https://chrisjmccormic.wordpress.com/2013/05/07/gradient-vectors/>. [Accessed 5 September 2015].

- [122] R. Lienhart, A. Kuranov and V. Pisarevsky, “Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection,” in *Pattern Recognition: 25th DAGM Symposium*, Magdeburg, 2003.